Titre: Title:	Génération d'itinéraires de passagers dans un réseau de transport aérien
Auteur: Author:	Éric Parent
Date:	2011
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Parent, É. (2011). Génération d'itinéraires de passagers dans un réseau de transport aérien [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/552/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/552/
Directeurs de recherche: Advisors:	Guy Desaulniers
Programme: Program:	Mathématiques appliquées

UNIVERSITÉ DE MONTRÉAL

GÉNÉRATION D'ITINÉRAIRES DE PASSAGERS DANS UN RÉSEAU DE TRANSPORT AÉRIEN

ERIC PARENT
DÉPARTEMENT DE MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION DU DIPLÔME DE
MAÎTRISE ÈS SCIENCES APPLIQUÉES
(MATHÉMATIQUES ET DE GÉNIE INDUSTRIEL)
MARS 2011

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

GÉNÉRATION D'ITINÉRAIRES DE PASSAGERS DANS UN RÉSEAU DE TRANSPORT AÉRIEN

présenté par : M. PARENT Eric, B.Sc.
en vue de l'obtention du diplôme de : Maîtrise ès Sciences Appliquées
a été dûment accepté par le jury constitué de :

M. SOUMIS François, Ph.D., président.
M. DESAULNIERS Guy, Ph.D., membre et directeur de recherche.
M. DESROSIERS Jacques, Ph.D., membre.

À Julie,
pour son indéfectible support, sa patience et son amour...

Remerciements

Je souhaite remercier infiniment mon directeur, le professeur Guy Desaulniers. Son support moral et financier ont permis de rendre ce projet possible alors que ses encouragement et conseils avisés auront contribué à sa complétion ; il a toujours fait preuve d'une très grande disponibilité à mon égard. Chaque contact avec lui m'aura permis d'apprendre une multitude de choses. Merci aussi au professeur François Soumis pour avoir participé à plusieurs discussions durant les travaux de recherche.

Je souhaite aussi remercier François Lessard et Benoit Rochefort pour avoir bien voulu partager une partie de leur expertise informatique avec moi. Leurs expériences constituent une richesse inespérée pour toute personne ayant besoin de conseils dans leurs champs d'expertise.

Un merci très particulier à chacun des membres de ma famille pour leur soutien moral. Merci tout spécialement à Julie, Simon, Rosalie, Charles et Émilie ; j'estime que cette réussite est en partie la leur.

Résumé

L'optimisation des opérations liées au domaine du transport aérien vise à réduire les coûts d'exploitation et à maximiser les revenus. La réduction des coûts peut être améliorée par la résolution de divers problèmes reliés à l'affectation des ressources comme la rotation des équipages par exemple. Les revenus peuvent être évalués par une approximation des ventes de billets et du positionnement par rapport à la concurrence et aussi grâce à la valeur ajoutée au service.

Dans le transport aérien, l'évaluation des revenus issus de la vente de billets est faite grâce à des modèles de flots de passagers. Ces modèles servent à déterminer de quelle manière un réseau de transport pourra fournir le service de transport à d'éventuels passagers et quels seront les niveaux d'achalandage des différents vols ; des niveaux élevés pourraient indiquer qu'une offre de service est insuffisante et suggérer aux planificateurs qu'une modification à l'horaire des vols soit requise ou qu'un type d'avion capable de transporter beaucoup de passagers est nécessaire pour suffire à la demande. Mais pour fonctionner, un modèle de flots de passagers requiert de connaître des itinéraires que les passagers sont susceptibles d'emprunter. L'objet des présents travaux de recherche porte sur des méthodes d'énumération d'un ensemble d'itinéraires de passagers.

Deux méthodes d'énumération sont présentées, toutes deux faisant appel à une approximation utilisée pour accélérer les calculs en limitant le nombre d'itinéraires générés. La première méthode se base sur un modèle mathématique fort répandu dans le domaine de l'optimisation des réseaux de transport : un réseau espace-temps. On y applique une variante de l'algorithme classique de recherche en profondeur (*depth-first search*). La seconde méthode est de nature combinatoire et ne requiert aucun modèle mathématique particulier mais fait appel à des structures de données accessoires afin de produire des combinaisons rapidement. Dans chaque cas, des variantes sont utilisées visant à améliorer le temps de calcul par une heuristique. L'heuristique utilisée est l'information relative aux bornes inférieures de temps entre chaque paire (O, D) où O est une origine et D une destination pour lesquelles on souhaite trouver des itinéraires. Dans un premier temps, les bornes sont calculées *a priori* et dans un second temps, elles sont mises à jour dynamiquement par une technique de relaxation.

Les techniques d'optimisation des réseaux de transport sont telles que les sous-problèmes, comme la rotation de personnel ou la confection d'un horaire des vols par exemple, soient résolus de manière itérative. Ainsi, des améliorations successives peuvent être apportées aux solutions de chacun de ces sous-problèmes jusqu'à l'obtention d'une solution globale jugée satisfaisante dans son ensemble. L'énumération d'itinéraires de passagers dépend d'un horaire de vols et doit être mise à jour dès lors qu'une modification y est apportée. Or ces modifications sont souvent faites localement, c'est-à-dire que seuls quelques vols sont modifiés, souvent n'affectant que quelques régions géographiques particulières. Il est donc intéressant de procéder uniquement aux re-calculs nécessaires et d'éviter les calculs redondants. La seconde méthode décrite ci-haut, dans sa version dynamique, sera mise à contribution dans de telles conditions.

La complexité temporelle de la méthode de recherche en profondeur est $O(n^2 \times d^{k_{max}})$ où m est le double du nombre de vols disponibles, k_{max} est le nombre maximum de vols que peut comporter un itinéraire de passager et d est le degré moyen d'un réseau construit à partir d'un horaire de vols. Pour la méthode combinatoire, on estime que sa complexité temporelle est $O(m^{k_{max}})$.

Les deux méthodes, à chaque fois dans ses deux variantes, sont soumises à des simulations numériques. Une liste d'itinéraires de passagers pour la saison d'été 2005 chez Air Canada est utilisée afin de représenter un contexte réel d'utilisation. On en extrait un horaire de vols qui servira de jeu de données. De plus, des informations complémentaires sont disponibles pour ces données comme le nombre de passagers par itinéraire. À partir de ces données, certains critères de performance peuvent être déduits comme la couverture de service des itinéraires générés par rapport aux itinéraires réels et le niveau de clientèle qui est mal desservi par l'horaire des vols, c'est-à-dire les paires (O, D) pour lesquelles le taux de couverture de service est en deçà de 85%. Les résultats numériques mettront en évidence que la méthode combinatoire avec mise-à-jour dynamique des bornes inférieures de temps de transport est la plus performante, tant aux points de vue des temps d'exécution et de certaines métriques sur l'ensemble d'itinéraires énumérés.

Les re-calculs partiels issus de modifications à un horaire de vols sont aussi réalisés et des temps de calculs sont comptabilisés pour diverses itérations. On note aussi le nombre d'itinéraires de vols qui ont dû être soit retirés, soit ajoutés à chaque itération.

Les critères de discrimination des itinéraires de passagers qui interviennent au mo-

ment de l'énumération sont propres à chaque clientèle et nous n'avons utilisé que des critères généraux. Mais ces critères, que chaque transporteur aérien considère comme de grande importance commerciale, peuvent être raffinés et factorisés. En outre, on peut aussi associer des paires d'origine et destination à des critères particuliers comme c'est le cas des stations balnéaires par exemple.

Abstract

Commercial airlines optimization aims at reducing costs and maximizing revenues. The cost reduction can be improved by solving various resource allocation problems such as crew pairing problem. The increase in revenues can be estimated by approximating passenger ticket sales, value-added services to customers compared to competition.

In air transportation networks, passenger flow models are used for evaluating revenues from passenger ticket sales. Such models are useful for determining how the network will provide services to customers and what are the occupation levels on each flight. High levels of occupation could indicate that a service offer is insufficient. This, in turn, could suggest to the planner that a flight schedule modification is needed or that reviewing the type of aircraft assigned to satisfy the demand is needed. However, to function well, the passenger flow model requires a set of passenger itineraries in input. The objective of this masters thesis is to present and compare methods for enumerating a set of passenger itineraries.

Two methods are introduced, both of them using a heuristic for accelerating computations by limiting the number of itineraries being generated that are not suitable from the passengers point of view. The heuristic in use is a lower bound on the travel time between each pair (O, D) of airports, O being an origin and D a destination. The first method uses a mathematical model vastly used in transportation networks optimization, a space-time network. We proceed with a modified version of a classical algorithm on graphs, the depth-first search. The second method is a combinatorial approach, leaves aside the need for the mathematical model used by the first method and only uses intermediate data structures for the fast retrieval of subsets of itineraries, which depends on the combinations to be evaluated. In each case, an alternate version is also implemented in regards to the computing of the heuristic's informations. In the original version of the two algorithms mentioned above, the heuristic is computed as a pre-treatment and values are obtained by a shortest-path algorithm whereas the heuristic is being updated dynamically in the alternate version by use of a relaxation technique.

Actual optimization techniques in use for transportation networks are such that

sub-problems, like crew assignment or flight scheduling for instance, are resolved in an iterative process. This leads to successively improving each solution of the sub-problems up to a point where a global, satisfying solution is found. The passenger route enumeration depends on the flight schedule and the resulting set needs to be updated whenever the schedule is modified. However, these modifications are often made locally, meaning that only a limited number of the passenger itineraries are being affected by the modifications. This, in turn, means that only a few geographical regions are being concerned. It is therefore interesting to proceed by performing only local re-computations, avoiding useless re-computations. And the second method described above, in its dynamic version, is being tested in such conditions.

Time complexity of the depth-first method is estimated to be $O(n^2 \times d^{k_{max}})$ where n is twice the number of available flight legs, k_{max} is the maximum number of flight legs allowed in a passenger itinerary and d is the average degree of a network constructed from a flight schedule. For the combinatorial method, the complexity is $O(m^{k_{max}})$.

Numerical simulations are performed with both methods and on each version, which are with and without pre-treatment for heuristic values. A list of passenger itineraries from Summer of 2005 for Air Canada customers is being used as a data set, allowing for simulations with real-world conditions. From this list we extracted a flight schedule to be used as an input for the enumeration methods. Moreover, additional informations are available for each itinerary in the list, such as the number of passengers on board. Given this information, we can evaluate some performance criteria about the set of itineraries as enumerated by our methods like the ratio of service being addressed by our generated itineraries and the market share not being served appropriately. Results will demonstrate the combinatorial method with dynamic travel time lower bounds update to be the best in regards to the execution time and metrics on the enumerated itineraries.

Partial re-computations required after modifications of a flight schedule are performed and computation times are accounted for various successive flight schedule modifications. We also consider the number of affected itineraries, both the added or removed ones, at each iteration.

The acceptance criteria being involved while evaluating the itinerary enumeration process are specific to each market and we only used generic criteria. Moreover, the criteria can be associated with specific (O, D) in treatment; one could think of tourists resorts needs compared to those of commercial travelers. Such information

being considered of high commercial value among industrials, it is not disclosed to the world outside of a given airline company. Furthermost, one could map origin and destination pairs with particular criteria like those involving destinations towards resorts for example.

Table des matières

Liste des tableaux

Liste des figures

Chapitre 1

INTRODUCTION

Les compagnies aériennes cherchent des moyens d'optimiser leurs profits. Nous considérons ici les compagnies de transport aérien de passagers telles que, par exemple, Air Canada, British Airways et Delta Airlines. Les revenus de ces transporteurs proviennent essentiellement des billets vendus à des passagers. Les dépenses, quant à elles, sont liées aux employés, à l'utilisation d'avions et à d'autres frais opérationnels connexes comme les frais d'accès aux aéroports ou encore les primes d'assurances. Les revenus dépendent ultimement des horaires de vols qui sont offerts aux passagers. Le problème de maximisation des profits pour ces entreprises commerciales est donc un problème de gestion des opérations. De par sa nature, un tel problème est complexe. Pour être traité, il est souvent divisé en sous-problèmes qui sont résolus séquentiellement (Klabjan (2005)). Ces étapes sont les suivantes.

(i) confection de l'horaire des vols ;

(ii) affectation des types d'avion ;

(iii) construction des rotations d'avion ;

(iv) construction des rotations d'équipage ;

(v) confection des horaires mensuels des membres d'équipage.

Ces étapes sont décrites dans les paragraphes suivants.

Le problème de *confection de l'horaire des vols* (i) concrétise l'offre de service que la compagnie aérienne souhaite offrir à une clientèle cible. Cette étape est cruciale dans l'approximation des revenus de l'entreprise. Cette offre de service, qui est cyclique sur une semaine pour presque toutes les semaines d'une saison, mènera vraisemblablement à des ventes de billets et procurera un revenu au transporteur. Des études de marchés sont préalablement faites afin d'en déterminer le contexte, que ce soit la compétition ou les opportunités commerciales. En général, ce travail de planification a lieu deux fois par année, soit une fois pour l'horaire d'été et une autre pour l'horaire d'hiver. De plus, des transitions sont considérées afin de pouvoir passer d'un horaire à l'autre de

manière graduelle. Un horaire de vols représente un ensemble de vols reliant diverses villes entre elles et où chaque décollage et atterrissage suit un certain horaire. Par exemple, dans le but de servir une clientèle de gens d'affaire, il pourrait être décidé d'offrir un service entre deux villes données seulement en début et fin de journée, durant les jours ouvrables de la semaine alors que d'autres clientèles pourraient plutôt requérir une offre de service uniquement durant la fin de semaine.

Un vol, ou segment de vol, est la donnée d'un décollage d'un avion à partir d'un aéroport donné et à un instant donné qui sera suivi d'un atterrissage à un aéroport de destination à un instant donné. Un billet vendu à un passager représente un itinéraire pour celui-ci. Le passager empruntera un ou plusieurs segments de vol pour effectuer son trajet. Avec une approximation du nombre de places occupées pour un segment de vol, les planificateurs pourront donc en estimer le revenu associé. Le revenu issu de la vente d'un billet peut ensuite être décomposé par segment de vol via une heuristique. Les coûts d'opérations sont connus pour chaque type d'avion dont l'équipage est déterminé, tout comme les coûts d'exploitation de l'avion, du carburant et des routines de maintenance. Le coût d'un billet varie par classe de transport ; on entend par classe de transport entre autres, la classe affaire, la classe économique ou la première classe.

Pour résoudre correctement le problème (i), il faut que celui-ci soit considéré conjointement avec le problème d'affectation des types d'avion (ii). En effet, il se pourrait qu'il n'existe pas d'affectation de types d'avion aux vols d'un horaire donné par cause de manque d'avions d'un certain type. Il peut aussi s'avérer économiquement judicieux de modifier une affectation des types d'avion à un vol pour en adapter la capacité de passagers selon le flot estimé, d'où l'utilité d'un modèle de flot de passagers, soit un modèle qui permet d'estimer le nombre de passagers attendus sur chaque itinéraire de passagers. La figure 1.1 présente les interactions ainsi envisagées entre ces problèmes ainsi que la séquence de résolution actuellement décrite.

Un modèle de flot de passagers requiert une liste d'itinéraires potentiellement intéressants pour des passagers de même qu'une affectation de types d'avion aux vols. Cette affectation de types d'avion permet d'établir une capacité en termes du nombre de passagers qui peuvent transiger pour chacun des segments de vol. On fait appel à un générateur d'itinéraires pour énumérer un ensemble d'itinéraires potentiellement intéressants pour les voyageurs. Un tel modèle de flot peut être résolu par une méthode de point fixe, comme c'est le cas dans Dumas et Soumis (2008) et il en résultera un nombre de places utilisées pour chaque segment de vol. Si le flot de passagers pour

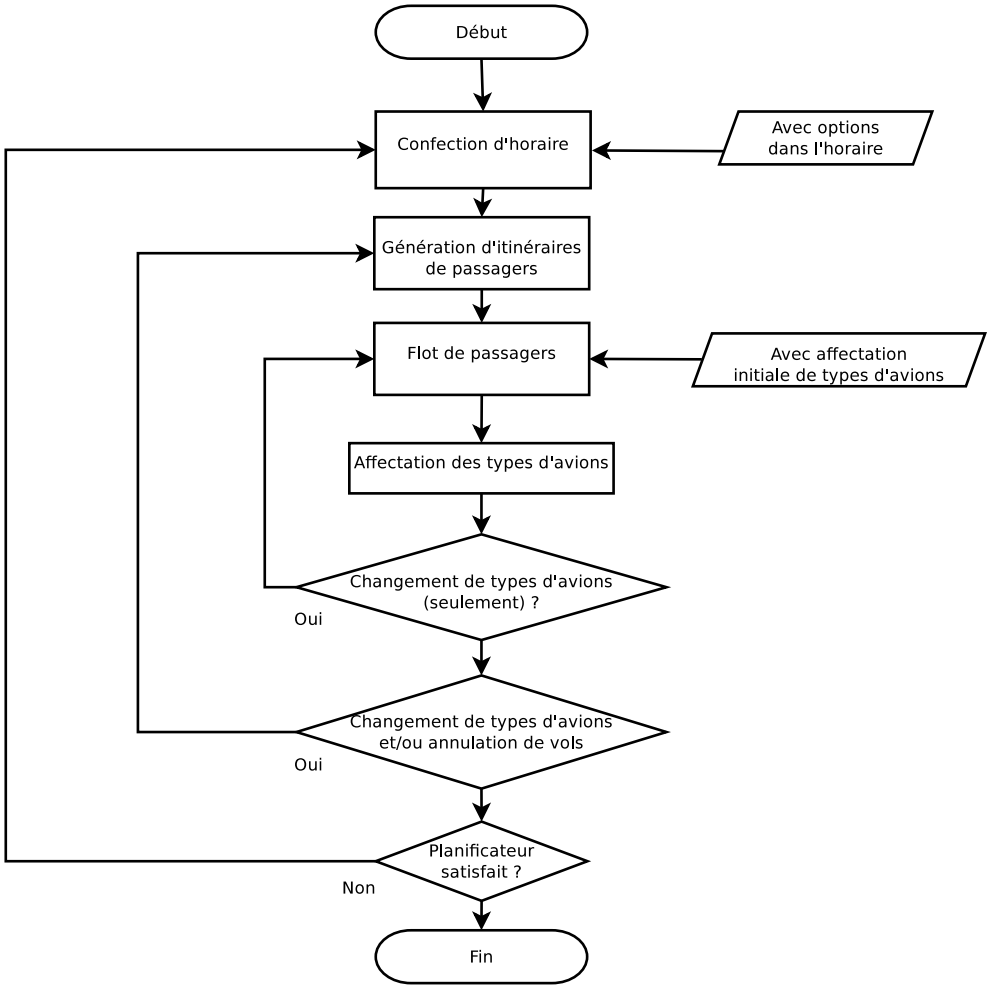

FIGURE 1.1 Décomposition en sous-problèmes avec rétroactions

un segment de vol donné est saturé et aurait pu être plus élevé, l'affectation du type d'avion pourrait être revue afin d'accommoder une clientèle plus nombreuse et, idéalement, augmenter les revenus.

Le problème d'*affectation des types d'avion* (ii) consiste à assigner un type d'avion à chaque vol décidé dans l'étape précédente. La principale contrainte dans ce problème est la quantité finie d'avions par type qui est à la disposition du transporteur. Dès lors que des avions sont affectés à des vols, il faut s'assurer que les avions de chaque type puissent couvrir tous les segments de vol qui leur sont affectés sans faire des déplacements à vide, tout en tenant compte des temps d'inutilisation dûs aux entretiens obligatoires. On parle alors du problème de *construction des rotations d'avion*

avec contraintes d'entretien (iii). Les entretiens varient d'une compagnie aérienne à l'autre selon la législation qui s'applique. De plus, ces contraintes varient d'un type d'avion à un autre. Les normes en vigueur prennent habituellement en compte les données historiques par avion et par compagnie aérienne afin d'assurer un suivi plus strict auprès des avions et transporteurs ayant présenté le plus d'anomalies dans le passé.

Pour qu'ils puissent fonctionner et assurer un service de qualité, les avions doivent être opérés par un équipage composé de pilotes, navigateurs et agents de bord. Il faut s'assurer que ces employés puissent revenir à la maison périodiquement ; on doit donc résoudre le problème de *rotations d'équipage* (iv). De plus, des contraintes additionnelles s'appliquent, que ce soit au niveau des conditions de travail réglementées, de conventions collectives ou des niveaux de compétence des divers postes. En effet, ce ne sont pas tous les pilotes qui sont habiletés à piloter n'importe quels types d'avion. Des considérations analogues existent au niveau des agents de bord. Lorsque chacun des problèmes mentionnés ci-haut sont résolus, il faut faire la *confection d'horaires mensuels* (v) pour les équipages. À cette étape, chaque membre d'équipage se voit attribuer un horaire composé de rotations, de jours de congés, et possiblement de périodes d'entraînement et de vacances. Ces horaires peuvent être construits de façon anonyme ou personnalisée en prenant en compte des activités pré-affectées aux employées et leurs préférences.

Dans ce mémoire, nous nous intéressons au problème d'énumération des itinéraires dans le but d'alimenter un modèle de flots de passagers. Il faut cependant veiller à ne pas énumérer trop d'itinéraires car ceci aurait pour effet de ralentir considérablement l'exécution d'un modèle de flots de passagers. Nous cherchons conséquemment à trouver une méthode qui soit rapide afin d'être utilisable dans un contexte de confection des horaires de vols parce qu'elle sera sollicitée lors de multiples itérations afin de s'assurer d'obtenir une solution réalisable et efficace. On peut donc penser à des méthodes de programmation dynamique avec des stratégies d'accélération.

L'objet du dénombrement concerne les *itinéraires de passagers*. Chacun de ces éléments consiste en une suite d'événements qui surviennent dans le temps et qui permettent à un voyageur de partir d'un aéroport d'*origine* et de se rendre à un aéroport de *destination*. Cet itinéraire peut être *direct*, c'est-à-dire qu'il peut se rendre de l'origine vers la destination sans effectuer d'atterrissage intermédiaire. Si ce n'est pas le cas, on dira alors qu'il est *composé*. Dans ce cas, on peut avoir un itinéraire avec

une ou plusieurs *correspondances* ou avec une ou plusieurs *escales*. Une *escale* est le cas de figure où un passager est dans un avion qui atterrit dans un aéroport intermédiaire pendant un temps restreint et redécolle plus tard sans qu'il n'ait eu à changer d'avion. S'il y a changement d'avion, et donc de vol, on parle plutôt d'une *correspondance*. La figure 1.2 présente un exemple d'itinéraire avec correspondance partant de Vancouver, dont le code d'identification attribué par l'agence internationale de transport aérien (AITA) est YVR, vers Halifax (code YHZ) ; le premier segment de vol se fait par le vol AC 123 alors que le second segment de vol s'effectue par le vol AC 417. La correspondance s'effectue à Edmonton (YEG).

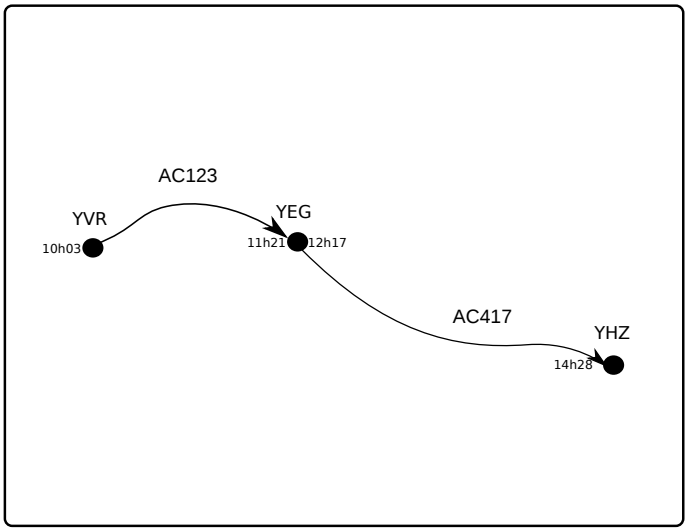

FIGURE 1.2 Exemple d'itinéraire entre Vancouver (YVR) et Halifax (YHZ) avec correspondance à Edmonton (YEG)

Lorsque vient le moment de comparer des itinéraires entre eux, il faut pouvoir les ordonner et nous utilisons pour ce faire la notion de *désagrément* d'un itinéraire. Ici, la notion de *désagrément* est une mesure plus ou moins abstraite du degré d'inconfort, moral ou physique, qu'un passager associe à un itinéraire. On dira qu'un événement ou un itinéraire est *désagréable* s'il augmente le désagrément d'un passager. Remarquons que cette notion est relative. Par exemple, si un itinéraire d'une durée de deux heures existe entre un aéroport A vers un aéroport B, il peut être jugé désagréable par rapport à un second itinéraire d'une durée moindre. Inversement, si ce premier itinéraire s'avère être le plus rapide, alors il est de désagrément moindre

que d'autres nécessitant un temps plus long. De la même manière, un itinéraire avec escale est moins désagréable qu'un autre itinéraire avec correspondance où le passager doit changer d'avion ; survient alors le risque de manquer l'embarquement dans le second avion si le premier atterrissage s'effectue avec du retard, tout comme le risque de perte de bagages. Donc, la durée totale du vol par rapport au vol le plus court est un facteur de désagrément, tout comme le nombre d'escales et de correspondances, une correspondance étant plus désagréable qu'une escale.

Les préférences des usagers du transport aérien sont nombreuses et diffèrent d'une clientèle à l'autre. Le but de ce mémoire est d'énumérer tous les itinéraires de passagers intéressants. Nous ne tenterons pas ici d'établir une hiérarchisation universelle des préférences des usagers mais plutôt d'utiliser les principaux critères de désagrément pour déterminer si un itinéraire est acceptable ou non.

Le reste de ce mémoire présente une revue de littérature dans le chapitre 2, une description de la problématique et de la modélisation dans le chapitre 3, suivi des algorithmes développés dans le chapitre 4 et des résultats numériques obtenus avec ceux-ci dans le chapitre 5. Finalement, une conclusion fait le point sur les travaux effectués et discute de directions de recherches futures par rapport à ce problème.

Chapitre 2

REVUE DE LITTÉRATURE

La littérature scientifique présente peu d'articles portant sur l'énumération d'itinéraires de passagers. En effet, le caractère confidentiel des données requises pour procéder à une telle génération de données touchant les opérations aériennes commerciales est sensible et à même de révéler les particularités des marchés visés par chacun des transporteurs. Pour le premier trimestre de 2009-2010 uniquement, Air France a annoncé un chiffre d'affaires de 5,19 milliards d'euros et pour l'exercice complet de 2008-2009, elle a déclaré un chiffre d'affaires de 23,970 milliards d'euros. Air Canada, quant à elle, a déclaré un chiffre d'affaires de 2,931 milliards de dollars canadiens pour son premier trimestre de 2009 et 11 082 millions de dollars canadiens pour l'exercice complet 2008.

Cette section présente une revue de littérature sur des sujets de recherche en rapport avec le problème qui nous intéresse. Notons que le domaine du transport en commun par autobus a fait l'objet de publications touchant la génération d'itinéraires de passagers. C'est l'application d'algorithmes visant la génération des k plus courts chemins dans un graphe qui a été utilisée dans ce cas. Nous commencerons donc par une revue de quelques algorithmes récents qui ont pour objectif de résoudre le problème des k plus courts chemins dans un graphe. Ensuite, on verra quelques publications sur la génération d'itinéraires de passagers dans le contexte d'un réseau de transport en commun. On poursuivra avec des travaux traitant des préférences des usagers dans le transport afin de déterminer les préférences générales des usagers lors qu'arrive le moment de déterminer si un itinéraire de passagers est intéressant par rapport aux autres. Finalement, nous parlerons brièvement d'un travail de recherche en lien direct avec le présent sujet de recherche, à savoir la génération de correspondances dans un réseau de transport aérien.

2.1 Chemin dans un graphe

Dans sa formulation générale, le problème de trouver les k plus courts chemins (kPCC) reliant deux sommets dans un graphe peut être résolu de manière exacte par des algorithmes s'exécutant en temps polynomial. Il en va de même pour le problème plus difficile de trouver les k plus courts chemins simples (c'est-à-dire sans cycle de longueur unitaire) dans un graphe. Ici, nous considérons un graphe G dont l'ensemble de ses n sommets est désigné par V, E désigne l'ensemble des m arêtes et k est le nombre de plus courts chemins à trouver. De tels algorithmes sont décrits dans Eppstein (1998) mais aussi plus récemment dans Hadjiconstantinou et Christofides (1999) et dans Gotthilf et Lewenstein (2009).

Pour le cas des plus courts chemins admettant des cycles, notons que l'implémentation décrite dans Eppstein (1998) s'applique aux réseaux, ou graphes orientés et dont les arcs sont pondérés (voir Labelle (1981)), pour lesquels on admet des arcs multiples et des *boucles* ; une boucle étant un arc dont le point de départ et le point d'arrivée sont confondus. L'algorithme qui est décrit dans Eppstein (1998) permet de trouver les k plus courts chemins d'un sommet initial s vers tous les autres sommets et s'exécute en $O(m + n\log(n) + kn)$. Une utilisation de cette implémentation permettrait de trouver tous les plus courts chemins entre chaque paire de sommets avec une complexité temporelle de $O(nm + n^2(\log(n) + k))$ alors que Hadjiconstantinou et Christofides (1999) décrit une implémentation dont la complexité temporelle est $O(kn^2)$. Remarquons que les chemins avec cycles sont de peu de valeur dans un contexte de transport.

Gotthilf et Lewenstein (2009) décrivent un algorithme pour trouver les k plus courts chemins sans cycle. Cette méthode est caractérisée par une complexité temporelle qui est $O(k(mn+n^2\log(\log(n))))$ en faisant un usage judicieux d'un algorithme permettant de trouver les plus courts chemins entre toutes paires de sommets d'un graphe, ou *all pairs shortest paths (APSP)*. On y utilise une notion de «détour» d'un chemin où, à partir du plus court chemin $P_{s,t}$ entre s et t, on cherche un détour D dans ce chemin qui augmente le poids du chemin mais de manière minimale. L'algorithme utilisé pour résoudre le problème APSP dans cette méthode est celui décrit par Pettie (2004) et s'exécute en $O(mn + n^2\log(\log(n)))$.

Le problème classique de trouver un plus court chemin dans un graphe est résolu par des algorithmes s'exécutant en temps polynomial sur la taille du graphe. Ces al-

gorithmes font partie des ouvrages d'introduction à l'algorithmique tels que Aho *et al.* (1974) et Cormen *et al.* (2001) ; notons l'algorithme de Dijkstra pour lequel certaines implémentations s'exécutent en $O(n^2)$. Ces algorithmes sont exacts et reposent sur des propriétés de graphes.

Le domaine de l'intelligence artificielle a développé un grand nombre de méthodes visant à accélérer la recherche d'un chemin dans un graphe entre deux sommets. Citons par exemple Russell et Norvig (1995), Dechter et Pearl (1985) et Koenig *et al.* (2004). On compte entre autres l'algorithme A* qui effectue une recherche dans un graphe de type *best-first* où le premier meilleur chemin constitue une solution au problème. Il utilise une évaluation de la distance restante pour déterminer si le chemin à emprunter semble valide ou non. L'heuristique consiste donc à additionner le coût du chemin parcouru jusqu'à date et une estimation du coût restant. On évalue le coût total du chemin reliant s à t par une fonction $f(s,t) = g(s,u) + h(u,t)$ où g est le coût du chemin emprunté du sommet de départ jusqu'au sommet actuel u, et h donne une borne inférieure du coût du chemin du sommet u vers le sommet de destination t. Dans le cas d'algorithmes cherchant un chemin dans un espace euclidien, comme un chemin géographique par exemple, c'est habituellement la norme euclidienne qui est utilisée pour h. Or, l'heuristique h présuppose un minimum de connaissances *a priori* au sujet de l'espace de recherche.

Une autre heuristique qui semble donner de bons résultats en moyenne est la recherche bidirectionnelle. Les méthodes dites unidirectionnelles, comme celle de Dijkstra, effectuent la recherche en partant d'un sommet s et procèdent avec une recherche de chemin vers le sommet t. La recherche bidirectionnelle effectue une recherche à la fois du sommet s vers t et de t vers s. Dans Luby et Ragde (1989), on voit que le comportement moyen de la recherche bidirectionnelle est concluante plus rapidement que la méthode unidirectionnelle. En effet, pour un graphe ayant n sommets et dont les arêtes ont toutes des poids positifs, on obtient un temps d'exécution de $\Theta(n \log(n))$ pour une implémentation de Dijkstra utilisant des monceaux alors que la méthode bidirectionnelle s'exécute plutôt en $\Theta(\sqrt{n} \times \log(n))$. L'article de Kaindl et Kainz (1997) fait une revue en profondeur de cette heuristique.

2.2 Itinéraires de passagers

Le problème de confection d'itinéraires de passagers dans le transport en commun a été traité par génération de plus courts chemins dans un réseau. Divers algorithmes ont été développés et appliqués, notamment par Wu et Hartley (2004) et aussi par van der Zijpp et Catalano (2005).

Dans le cas de la génération d'itinéraires de passagers dans un réseau de transport bi-modal, c'est-à-dire autobus et marche, Wu et Hartley (2004) utilisent un algorithme des k plus courts chemins (kPCC). Dans cet article, les auteurs procèdent en calculant les k plus courts chemins dans un réseau pour ensuite éliminer les chemins ne satisfaisant pas les conditions attendues, comme un nombre maximum de transferts entre les lignes d'autobus par exemple, ou encore une distance maximale parcourue à pieds. Une telle méthode peut requérir un temps d'exécution plutôt long si peu d'itinéraires parmi les k trouvés initialement satisfont les critères attendus. Par exemple, si nous avons k itinéraires trouvés initialement et que nous en retranchons p, nous avons donc $k - p = m$ itinéraires restants qui satisfont les critères de sélection. Si nous cherchons en fait à trouver M chemins et que $m < M$, il faudra alors augmenter la valeur de k en espérant augmenter la liste de chemins puis retrancher ceux qui font défaut. On doit répéter ce procédé jusqu'à l'obtention de $m \geq M$. Cette méthode procède donc par retranchement des solutions qui ne sont pas valides. Des résultats de simulations numériques qui y sont présentées montrent que trouver les k plus courts chemins dans un réseau avec 2 398 arrêts d'autobus prend un temps de calcul de l'ordre de 2 secondes pour $k = 2$ et de l'ordre de 50 secondes pour $k = 4$, selon la longueur des itinéraires, en nombre d'arrêts d'autobus. Ces calculs visent une seule paire (origine, destination) à la fois.

Dans van der Zijpp et Catalano (2005), on compare la méthode des k plus courts chemins avec une méthode de résolution du problème des k plus courts chemins avec contraintes. Ceci vise principalement à éliminer le problème de calculs successifs de chemins en faisant varier la valeur de k après avoir retranché p solutions invalides. L'astuce adoptée cherche à trouver les k plus courts chemins qui satisfont *a priori* un certain nombre de contraintes définies initialement. Pour cela, les auteurs adaptent un algorithme (Lawler (1976)) qui calcule les k plus courts chemins dans un réseau acyclique. Cet algorithme initial consiste en une méthode de séparation-et-évaluation (*branch-and-bound*) et est donc défini récursivement. La stratégie principale décrite

dans Lawler (1976) consiste à procéder en effectuant un partitionnement de l'ensemble des chemins dans le réseau. À partir du plus court chemin dans le réseau, on force un partitionnement pour ensuite chercher un plus court chemin dans chacune des parties. Le partitionnement reprend pour chacune des sous-parties jusqu'à trouver k chemins. Cet article compare donc deux types d'algorithmes pour résoudre le problème kPCC sous contraintes, soit 1) la méthode de retranchement et 2) le calcul direct des k plus courts chemins qui respectent des contraintes (kPCCSC). Pour pouvoir appliquer une astuce de type *branch-and-bound*, la notion de contrainte définie récursivement y est présentée. En effet, si nous avons un chemin c_1 dans lequel une contrainte n'est pas respectée, alors il en ira de même pour tout chemin C qui est la concaténation du chemin c_1 avec un chemin c_2. Les résultats numériques présentés par van der Zijpp et Catalano (2005) montrent que générer les k plus courts chemins sous contraintes pour un graphe de 400 sommets prend un temps de calcul de l'ordre de la minute.

2.3 Préférences des usagers dans le transport

Le choix d'un usager du transport aérien pour un transporteur est guidé par divers critères. Certains critères décisionnels sont énumérés dans Proussalogloua et Koppelman (1999) et Wen et Lai (2009). On effectue une pondération sur chacun d'entre eux afin de les relativiser. De manière analogue, le choix d'un itinéraire par un passager dépendra de critères les classifiant relativement entre eux ; un itinéraire moins attrayant sera mis de côté par le passager si un autre itinéraire lui étant plus avantageux existe et est disponible. Par exemple, dans Baaj et Mahmassani (1995), il y est mentionné que des itinéraires dont le temps est 50% plus long que le temps du plus court chemin pour une origine et destination donnée sont des itinéraires inintéressants du point de vue de l'usager du transport en commun.

Quelques préférences des passagers du transport aérien ont été identifiés dans Grosche et Rothlauf (2007) : ces données et quelques autres constituent les entrées de divers modèles pour évaluer la part de marché relative des itinéraires pour divers transporteurs aériens. Entre autres, les préférences d'usagers portent sur 1) le type d'un itinéraire, soit s'il est direct ou non ; 2) le rapport entre le temps d'un itinéraire et le temps du plus rapide itinéraire reliant les mêmes origine et destination ; 3) le type de l'itinéraire le plus court reliant les mêmes origine et destination.

Dans Proussalogloua et Koppelman (1999), puis dans Coldren *et al.* (2003), un

modèle *logit* multinomial est étudié pour déterminer le degré d'intérêt relatifs des itinéraires entre les transporteurs. Entre autres, on y considère les critères de 1) *niveau de service* d'un itinéraire, c'est-à-dire s'il est direct, avec un arrêt ou deux arrêts en comparaison avec le meilleur itinéraire pour une paire origine et destination donnée ; 2) le rapport entre les distances d'un itinéraire visé et le meilleur itinéraire pour une origine et destination donnée ; 3) la différence entre le temps de l'itinéraire visé et celui de l'itinéraire le plus rapide pour une origine et destination donnée ; 4) le nombre de sièges disponibles dans l'avion ; 5) l'heure de départ de l'itinéraire. D'autres critères plus spécifiques à la compagnie aérienne sont aussi utilisés, notamment la présence d'un kiosque du transporteur dans l'aéroport de départ et d'arrivée. De plus, Koppelman *et al.* (2008) indique que les préférences des usagers du transport aérien sont différentes selon que l'usager se trouve à l'étranger et désire rentrer au pays ou s'il est plutôt dans le cas contraire.

2.4 Correspondances dans un réseau aérien

Des travaux antérieurs ont été faits par Lebeau (1984) qui a travaillé sur le problème de génération de correspondances de passagers dans un réseau de transport aérien sur un horizon de temps d'une journée. On y trouve une méthode qui tient compte de données de marchés des clientèles. Un *marché* est défini par une demande de transport entre deux aéroports pour une plage horaire donnée dans une journée. La demande est exprimée en nombre de passagers. Par exemple, les clients partant de Montréal en direction de Toronto entre 07h30 et 08h30 pour chacun des jours du lundi au vendredi représentent une clientèle de gens d'affaires.

La méthode développée par Lebeau (1984) consiste en une adaptation de l'algorithme de Floyd-Warshall qui tient compte d'une heuristique admissible comme celle utilisée dans l'algorithme A* pour deux aéroports u et v et où l'heuristique donne la distance à vol d'oiseau entre ces deux aéroports. La méthode tient compte aussi de la répartition des passagers entre les divers vols d'un marché vers un autre. Des approximations probabilistes de débordement et de réaffectation (*spill and recapture*) sont faites et un coût est associé à de telles opérations afin de représenter une insatisfaction perçue par les passagers devant choisir un chemin alternatif d'un autre marché que celui qui est normalement leur marché. En affectant un calcul sommaire de flot de passagers au moment de générer un itinéraire, ou une correspondance pour

reprendre le terme que l'auteure emploie, on utilise les notions de flots de passagers pour éviter de générer des itinéraires n'apportant aucune amélioration à la couverture de service et, ultimement, de revenus.

On permet donc d'associer un coût à chaque correspondance qui représente une insatisfaction du passager. Les correspondances représentant une insatisfaction plus grande qu'un certain seuil ne sont pas conservées.

Chapitre 3

PROBLÉMATIQUE

Le présent chapitre formule clairement la problématique qui doit être résolue. On y présente la terminologie et les définitions employées dans le reste de ce mémoire, tout d'abord celles reliée aux itinéraires de passagers et ensuite celle en rapport avec la notion de désagrément associée à un itinéraire. Nous présentons finalement le modèle utilisé pour représenter le problème à résoudre.

3.1 Définitions et terminologie

L'objectif principal abordé dans ce mémoire consiste à énumérer les itinéraires qui sont susceptibles d'intéresser un usager d'un réseau aérien visé. Afin d'arriver à cette fin, il faut pouvoir :

1. trouver des itinéraires dans un réseau de transport aérien ;
2. définir des critères d'insatisfaction relativement à des itinéraires du point de vue des usagers.

Les critères d'insatisfaction seront utiles lorsque nous chercherons à savoir si un itinéraire trouvé doit être conservé.

3.1.1 Itinéraire de passagers

Voici quelques définitions qui permettront de décrire formellement ce qu'est un itinéraire de passagers. Nous voyons principalement les entités qui sont définies dans un horaire de vols, en commençant bien naturellement avec une définition de ce qu'est un vol.

Considérons, à partir de maintenant, que l'ensemble des aéroports visités par un transporteur aérien est noté par $\mathbb{A}$.

Definition 3.1.1. Un *segment de vol* ou *vol*, $S = ((A, t_A), (B, t_B))$, est un déplacement d'un avion constitué d'un décollage survenant au temps t_A suivi d'un seul atterrissage survenant au temps t_B reliant un aéroport A à un aéroport B où $A \neq B$.

Nous avons le tuple $(A, t_A) \in \mathbb{A} \times \mathbb{T}$ où $\mathbb{T}$ est l'ensemble des jours et heures de la semaine.

Les segments de vol représentent donc une portion insécable du transport aérien car il s'agit d'un décollage d'avion à un aéroport d'origine suivi de l'atterrissage de ce même avion dans un aéroport de destination.

Definition 3.1.2. On considère par *marché* une demande visant une paire d'aéroports à un ou plusieurs moments de la semaine. Un marché est associé à une clientèle cible.

Par exemple, on considère que les gens d'affaires voulant se rendre de Montréal à Toronto à 08h30 du lundi au vendredi représente un marché.

Definition 3.1.3. Un ensemble de vols constitue un *horaire de vols* et représente une offre globale de service de la part d'un transporteur aérien visant à satisfaire certaines parts de marché.

Afin d'assurer une régularité dans la planification et pour faciliter le travail de planification des opérations, les compagnies aériennes construisent un horaire de vols qui est cyclique sur une semaine. Nous posons donc que le jour 1 est le lundi, le mardi est le jour 2 et ainsi de suite jusqu'au jour 7 qui représente le dimanche. Cette suite étant cyclique, on a que le jour qui succède au dimanche (jour 7) est le lundi (jour 1). Étant donné la nature des semaines consécutives, il est possible qu'un vol partant d'un aéroport d'origine le dimanche (jour 7) vers 23h00 pourrait arriver à un aéroport de destination le lundi (jour 1) à 02h15, par exemple.

Un horaire de vols est valable pour chacune des semaines ayant lieu dans une saison donnée mais admet des exceptions. Ces évènements qui débordent de la planification saisonnière sont traités spécifiquement comme, par exemple, une offre de service particulièrement élevée lors de situations comme des crises humanitaires ou de gros évènements culturels ou sportifs tels que la tenue de jeux olympiques par exemple. Certains déplacements additionnels d'avions nécessaires pour des considérations opérationnelles, comme l'entretien ou la rotation de types d'avions, peut amener les planificateurs à ajouter momentanément des vols et ainsi éviter, ou à tout

le moins réduire, les frais reliés à de telles opérations. De même, on peut devoir introduire graduellement de nouveaux vols grâce à de nouveaux avions achetés dans le but d'augmenter une flotte existante ou d'en remplacer une partie.

Definition 3.1.4. Un *itinéraire de passager* $I = (S_1, S_2, \ldots, S_j)$ est une suite finie et ordonnée dans le temps de segments de vol qui sont empruntés par un passager pour lui permettre de se rendre d'un aéroport d'origine vers un aéroport de destination.

Definition 3.1.5. Une *escale* consiste en un arrêt dans un aéroport intermédiaire où le passager n'a pas à changer d'avion.

Definition 3.1.6. Une *correspondance* consiste en un arrêt dans un aéroport intermédiaire où le passager doit changer d'avion.

Un itinéraire de passager peut être de l'un ou l'autre de ces types :

1) direct : l'itinéraire ne connaît qu'un seul décollage, soit celui à l'aéroport d'origine, et un seul atterrissage, soit à l'aéroport de destination ;
2) avec une ou plusieurs escales ;
3) avec une ou plusieurs correspondances ;
4) composés d'escale(s) et de correspondance(s).

Soit $\mathbb{S}$ l'ensemble de tous les segments de vol définis pour un transporteur aérien. Définissons deux fonctions qui s'appliquent sur un segment de vol comme suit :

1. $\mathcal{O} : \mathbb{S} \to \mathbb{A}$ qui donne l'aéroport d'origine d'un segment de vol.
2. $\mathcal{D} : \mathbb{S} \to \mathbb{A}$ qui donne l'aéroport de destination d'un segment de vol.

Ces deux fonctions nous serons utiles pour définir la propriété de *connectabilité* de deux segments de vol que voici.

Definition 3.1.7. Par définition, on dira que deux segments de vol $S_1 = ((A, t_{A'}), (B, t_B))$ et $S_2 = ((B, t_{B'}), (C, t_C))$ sont *connectables* si et seulement si les conditions suivantes sont respectées :

(i) $\mathcal{D}(S_1) = \mathcal{O}(S_2)$;

(ii) $t_{max} \geq t_{B'} - t_B \geq t_{min}$, pour certaines valeurs $t_{max} > t_{min} > 0$.

On dira de deux segments connectables qu'ils ont la propriété de *connectabilité*.

3.1.2 Notion de désagrément

Au moment de faire un choix dans une liste d'itinéraires disponibles, le passager choisira un itinéraire qui lui apportera le moindre désagrément. Certains itinéraires sont jugés intéressants dans la mesure où ils représentent un niveau de désagrément qui est tolérable. Les itinéraires présentant des caractéristiques au-delà de ce ou ces seuils de décision, seront tout simplement ignorés par le passager. Bien entendu, les critères de décision sont relatifs. Par exemple, un itinéraire représentant un court temps de voyage est intéressant mais son prix pourrait amenuiser, voire annuler cet effet. Or, la nature humaine étant ce qu'elle est, le seuil de tolérance au désagrément varie d'une personne à l'autre, tout comme la gamme de besoins à combler par un achat de billet d'avion varie d'un client à un autre. Le but de ce mémoire ne se situe pas au niveau de la modélisation des préférences des usagers du transport aérien ou de la prise de décision. Cependant, pour pouvoir énumérer des itinéraires de passagers, il nous faut pouvoir déterminer quelques critères d'acceptation ou de rejet d'un itinéraire. Voici les critères qui seront utilisés afin de déterminer si un itinéraire est *acceptable* ou non :

1. l'itinéraire est composé d'un nombre maximum de segments de vol ;
2. l'itinéraire respecte un temps minimum de connexion t_{min} entre chaque paire de segments de vol consécutifs ;
3. l'itinéraire ne comporte aucun temps d'attente entre chaque paire de segments de vol consécutifs dans un aéroport intermédiaire qui soit plus grand que t_{max} ;
4. le temps total requis pour voyager d'un aéroport d'origine O vers une destination D doit être au plus un multiple λ fois le temps du plus court trajet entre O et D, où $\lambda \in \mathbb{R}, \lambda \geq 1$.

Au moment de confectionner un horaire de vols, le coût d'un billet n'est pas encore déterminé et peut varier selon certaines conditions, comme par exemple des promotions ou l'ajustement des coûts de billets en rapport avec le contexte du marché et de la concurrence, etc. Ces conditions de marché particulières sont connues habituellement peu de temps avant le moment où le vol a lieu. Ainsi, nous ne pourrons considérer un prix associé à un billet d'avion comme critère entrant en jeu au moment de déterminer le désagrément associé à un itinéraire, même si c'est un critère important pour les usagers du transport. Il est cependant raisonnable de penser que les processus de mise en marché et de ventes mis en place par les transporteurs aériens visent à minimiser ce facteur auprès des consommateurs.

3.2 Réseau mathématique

Depuis le tout début de ce mémoire, nous utilisons l'expression « réseau de transport aérien » pour désigner l'ensemble des services de transport dispensés par une compagnie aérienne. Le lecteur pourra se douter que c'est en lien avec l'objet mathématique « réseau ». Et c'est en effet cet objet mathématique qui sera utilisé. La définition mathématique d'un réseau est un graphe orienté et pondéré, c'est-à-dire un graphe $G = (V, E, \omega)$ constitué de $|V| = n$ sommets et de $|E| = m$ arcs dont chacun est pondéré par la fonction ω. Cette fonction $\omega : E \to \mathbb{N}$ associe à chaque arc $e \in E$ un *poids* qui est un nombre à valeur dans les naturels ($\mathbb{N}$). Nous prendrons la minute comme unité de temps. Nous savons qu'une semaine comporte $7\,\text{jours} \times 24\,\frac{\text{heures}}{\text{jour}} \times 60\,\frac{\text{minutes}}{\text{heure}} = 10080$ minutes en tout. À chaque jour et heure de la semaine, nous pourrons associer un instant en minutes de la manière suivante :

$$T(j, h, m) = m + 60 \times h + (j - 1) \times 60 \times 24$$

où j est le numéro du jour allant de 1 à 7, h est l'heure de la journée allant de 0 à 23 et m correspond aux minutes de l'heure qui vont de 0 à 59 inclusivement.

Soit un segment de vol $S = ((A, t_a), (B, t_b))$ où A, B sont des aéroports et t_a, t_b des instants de la semaine. On peut naturellement en déduire un arc $\tilde{S}$ dont le poids est la durée du segment de vol en minutes. Ainsi, le poids d'un arc $\tilde{S}$ est donné par

$$\omega(\tilde{S}) \;=\; |t_b - t_a| \mod 10880$$

En prenant le segment de vol S, nous avons que les évènements donnés par (A, t_a) et (B, t_b) représentent, respectivement, le sommet de début et le sommet de fin de l'arc $\tilde{S}$.

Considérons les sommets représentant des évènements survenant à un aéroport donné, atterrissages et décollages confondus. En représentant ces évènements sous forme de liste triée selon le temps, on obtient une suite chronologique des évènements ayant lieu à un aéroport donné. Posons $L = ((A, t_0), (A, t_1), (A, t_2), \ldots, (A, t_r)$ une liste d'évènements survenant à l'aéroport A et où $t_0 < t_1 < t_2 < \ldots < t_r$. On peut relier entre eux toute paire de sommets consécutifs (A, t_{j-1}) et (A, t_j) pour $j \in \{1, 2, \ldots, r\}$ dans cette liste triée par un arc d'attente $e_{(j-1,j)} = ((A, t_{j-1}), (A, t_j))$. Chacun de ces arcs d'attente ont un poids égal au temps d'attente qu'il représente, soit

$t_{j-1} - t_j = \omega(e_{(j-1,j)})$. De plus, on ajoute un arc d'attente reliant entre eux le dernier évènement de la semaine et le premier évènement de la semaine à survenir en un même aéroport. Ceci est requis pour représenter la cyclicité d'un horaire hebdomadaire.

La figure 3.1 présente un exemple simple du modèle utilisé pour le problème de génération des itinéraires de passagers. Pour des fins de clarté, cet exemple comporte uniquement deux aéroports, soit l'aéroport international Lester B. Pearson de Toronto (YYZ) et Montréal-Trudeau (YUL). Dans cette figure, les segments de vol sont représentés par les arcs en traits pleins alors que les arcs d'attente sont représentés par des traits pointillés.

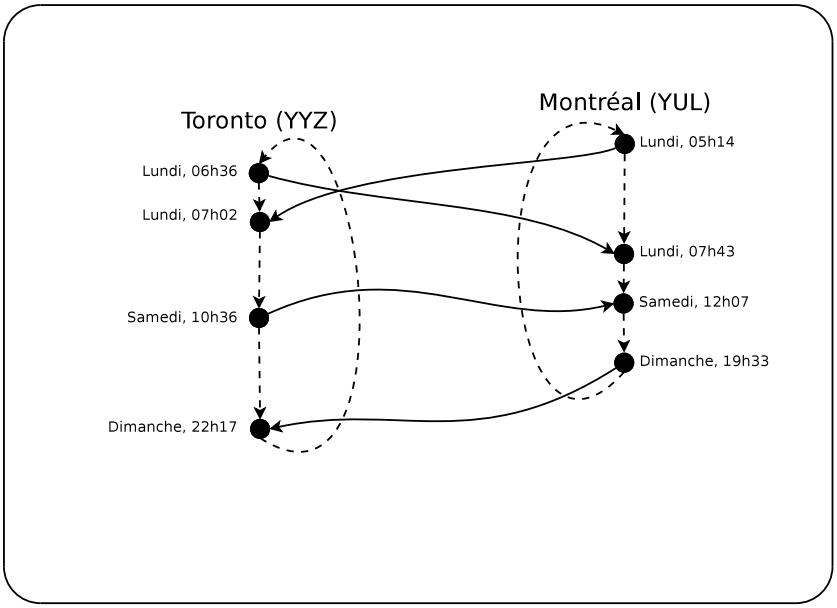

FIGURE 3.1 Réseau espace-temps

Remarque 1. Comme le réseau que nous venons de décrire représente des endroits (aéroports) et des instants (temps de décollage ou d'atterrissage), on pourra le désigner par *réseau espace-temps*.

En résumé, nous avons donc deux types d'arcs dans notre réseau, à savoir :

i) les arcs *explicites* qui représentent des segments de vol ;

ii) les arcs *implicites* qui représentent des temps d'attente entre deux évènements successifs.

Afin de mieux représenter la réalité, on dira que le temps d'atterrissage associé à un segment de vol représente l'instant où les roues de l'avion touchent le sol. Il faut donc considérer un délai qui correspond au temps requis pour que l'avion puisse se rendre de la piste d'atterrissage au débarcadère afin que les passagers puissent sortir de l'avion et débuter une connexion. Nous négligeons l'écart de temps qui sépare le débarquement entre le premier et le dernier passager d'un avion ; ce nombre dépend de la capacité de l'avion en terme du nombre de passagers, de sa configuration et ultimement du comportement des passagers lors du débarquement. De la même manière, le type d'avion détermine en partie le délai entre l'atterrissage et le début du processus de débarquement. En effet, ce ne sont pas tous les types d'avion qui peuvent atterrir sur n'importe laquelle des pistes d'atterrissage de n'importe quel aéroport. De plus, les aéroports dans le monde ne présentent pas tous la même configuration topologique et pour un type d'avion donné, le temps de déplacement entre la piste d'atterrissage et l'aire de débarquement peut varier. Nous négligerons cette variabilité par type d'avion et par aéroport. Le présent modèle tient compte du temps de déplacement de la piste au débarcadère de manière constante, peu importe le type d'avion et l'aéroport. Nous noterons ce temps de déplacement de la piste vers le débarcadère par une constante t_ϵ.

De manière analogue, nous négligeons la nature du vol, c'est-à-dire que nous ne faisons pas de distinctions entre le fait qu'il soit de portée internationale ou régionale. Les aéroports sont souvent construits de manière à ce que différentes parties de la superficie soient utilisées pour des vols de natures distinctes. Ceci permet de regrouper, entre autres, les services frontaliers qui doivent exercer leur juridiction dans le cadre de vols internationaux mais auxquels les vols régionaux ne sont pas soumis. De plus, les plus gros types d'avions utilisés pour les vols transcontinentaux requièrent des pistes d'atterrissage qu'il s'avère parfois utile, pour des considérations pratiques, de construire à des endroits différents des pistes pour les plus petits types d'avions habituellement utilisés pour les vols régionaux. Donc, la nature du vol pouvant avoir un impact sur le temps requis entre l'atterrissage et le débarquement des passagers. Cela peut aussi avoir un impact sur le temps requis par un passager pour se déplacer dans un aéroport.

Reprenons maintenant l'étape (i) qui associe un arc $\tilde{S}$ à un segment de vols donné $S = ((A, t_a), (B, t_b))$. Nous aurons maintenant un arc $\tilde{S} = ((A, t_a), (B, t_b))$ et le poids associé à cet arc sera $\omega(\tilde{S}) = t_b - t_a$.

3.3 Formulation du problème

Le problème qu'on cherche à résoudre consiste à énumérer l'ensemble des itinéraires de passagers entre chaque paire d'aéroports visée par une demande de service. Cet ensemble d'itinéraires permettra d'alimenter un modèle de flot de passagers. Mais il faut énumérer seulement les itinéraires correspondant aux critères d'acceptabilité présentés dans la section 3.1.2. Il est important de dire que nous allons procéder avec une énumération des itinéraires de passagers en lot, c'est-à-dire que chaque marché à être desservi commence au moins par un segment de vol présent dans l'horaire des vols. En effet, la confection d'un horaire de vols tient en compte les notions de marché et on obtient un horaire de vols qui est tributaire de ces requis. Les données de marché sont implicitement représentées dans le réseau mathématique utilisé pour représenter le contexte du problème puisqu'il sera confectionné à partir des données d'un horaire de vols. Si un marché se trouve mal desservi par l'horaire des vols, c'est le modèle de flot de passagers qui mettra ce fait en évidence.

Voyons maintenant la formulation formelle du problème à résoudre. Commençons tout d'abord par l'introduction de notations qui seront utilisées afin de préciser l'ensemble à énumérer explicitement.

Nous avons désigné par $\mathbb{A}$ l'ensemble de tous les aéroports qui sont desservis par un transporteur aérien. Introduisons ici une fonction qui renseigne sur la quantité de demande de transport associée entre deux aéroports :

$$\Delta : \mathbb{A}^2 \to \mathbb{R}$$

où, pour une paire d'aéroports (A_q, A_r), on obtient une valeur du nombre de personnes voulant voyager, à un moment ou un autre de la semaine, en partant de A_q vers A_r. Par définition, nous dirons que $\Delta(A_q, A_r) = 0$ lorsque $A_q = A_r$. Il semble tout à fait inutile, du point de vue du transport aérien, qu'une demande puisse exister entre un aéroport et lui-même. L'information fournie par la fonction Δ est de nature commerciale et permet de savoir s'il est pertinent ou non de vouloir énumérer des itinéraires de passagers entre deux aéroports donnés.

Désignons par $\mathbb{A}^2_*$ l'ensemble des paires d'aéroports pour lesquelles il y a une demande de transport aérien qui est non nulle, définit formellement comme suit :

$$\mathbb{A}^2_* = \{(A_q, A_r) \ : \ A_q \in \mathbb{A}, \ A_r \in \mathbb{A}, \ \Delta(A_q, A_r) > 0\} \tag{3.1}$$

Nous avons que $\mathbb{I}$ est l'ensemble des itinéraires entre chaque paire de villes dans $\mathbb{A}^2$. Nous posons $\mathbb{I}_*(A_q, A_r)$ l'ensemble de tous les itinéraires possibles partant de l'aéroport A_q et se rendant à l'aéroport A_r, pour $(A_q, A_r) \in \mathbb{A}^2_*$. Notons par $T_{INF}(A_q, A_r)$ le temps le plus court pour se rendre de l'aéroport A_q vers A_r.

Nous pouvons maintenant considérer l'ensemble

$$\mathbb{I}_* = \mathbb{I}(\mathbb{A}^2_*) = \bigcup_{(A_q,A_r)\ \in\ \mathbb{A}^2_*} \mathbb{I}(A_q, A_r) \tag{3.2}$$

de tous les itinéraires entre chaque paire d'aéroports pour lesquelles il y a une demande non nulle.

Voici quelques fonctions définies sur un itinéraire de passagers $I = (S_1, S_2, \ldots, S_j)$ composé de j segments de vol :

1. $\mathcal{L} : \mathbb{I} \to \mathbb{N}$ qui donne le nombre de segments de vol qui constituent un itinéraire ;
2. $\mathcal{A}_+ : \mathbb{I} \to \mathbb{N}$ qui retourne le plus grand temps d'attente, en minutes, dans un aéroport intermédiaire visité par l'itinéraire ;
3. $\mathcal{A}_- : \mathbb{I} \to \mathbb{N}$ qui donne le plus court temps d'attente, en minutes, dans un aéroport intermédiaire à survenir au courant d'un itinéraire ;
4. $\mathcal{T} : \mathbb{I} \to \mathbb{N}$ qui donne la durée totale d'un itinéraire exprimée en minutes ;
5. $\mathcal{O} : \mathbb{I} \to \mathbb{A}$ qui donne l'aéroport d'origine d'un itinéraire. Nous étendons ici une fonction utilisée dans la définition 3.1.7 ;
6. $\mathcal{D} : \mathbb{I} \to \mathbb{A}$ qui donne l'aéroport de destination d'un itinéraire. Nous étendons ici une fonction utilisée dans la définition 3.1.7.

De ces fonctions sur un itinéraire, nous pouvons obtenir des fonctions qui retournent un résultat de nature binaire lorsqu'elles sont pourvues d'un paramètre additionnel. Voici ce fonctions.

1. la fonction $\mathcal{L}^\bullet : \mathbb{I} \times \mathbb{N} \to \mathbb{N}$ est définie comme suit :

$$\mathcal{L}^\bullet(I, k_{max}) = \begin{cases} 1 & \text{si } \mathcal{L}(I) \leq k_{max} \\ 0 & \text{sinon} \end{cases}$$

2. la fonction $\mathcal{A}^{\bullet}_{+} : \mathbb{I} \times \mathbb{N} \to \mathbb{N}$ est définie comme suit :

$$\mathcal{A}^{\bullet}_{+}(I, t_{max}) = \begin{cases} 1 & \text{si } \mathcal{A}_{+}(I) \leq t_{max} \\ 0 & \text{sinon} \end{cases}$$

3. la fonction $\mathcal{A}^{\bullet}_{-} : \mathbb{I} \times \mathbb{N} \to \mathbb{N}$ est définie comme suit :

$$\mathcal{A}^{\bullet}_{-}(I, t_{min}) = \begin{cases} 1 & \text{si } \mathcal{A}_{-}(I) \geq t_{min} \\ 0 & \text{sinon} \end{cases}$$

4. la fonction $\mathcal{T}^{\bullet} : \mathbb{I} \times \mathbb{R} \to \mathbb{N}$ est définie comme suit :

$$\mathcal{T}^{\bullet}(I, \lambda) = \begin{cases} 1 & \text{si } \mathcal{T}(I) \leq \lambda \times T_{INF}(\mathcal{O}(I), \mathcal{D}(I)) \\ 0 & \text{sinon} \end{cases}$$

Nous pouvons, à l'aide de ces fonctions binaires, définir une fonction d'acceptation, qui est la conjonction de ces fonctions, comme suit :

$$\Pi : \mathbb{I} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{R} \to \{0, 1\}$$

où

$$\Pi(I, k_{max}, t_{min}, t_{max}, \lambda = \mathcal{L}^{\bullet}(I, k_{max}) \times \mathcal{A}^{\bullet}_{+}(I, t_{max}) \times \mathcal{A}^{\bullet}_{-}(I, t_{min}) \times \mathcal{T}^{\bullet}(I, \lambda) \quad (3.3)$$

La fonction Π nous permet de déterminer si un itinéraire trouvé peut, ou non, être ajouté dans l'ensemble à énumérer. Cette fonction doit être pourvue des paramètres $k_{max}, t_{min}, t_{max}, \lambda$ pour pouvoir être évaluée. Finalement, l'ensemble d'itinéraires à énumérer est formellement décrit comme suit :

$$\tilde{\mathbb{I}} = \{I \mid I \in \mathbb{I}_{*},\ \Pi(I, k_{max}, t_{max}, t_{min}, \lambda) > 0\} \quad (3.4)$$

Il est donc possible de caractériser l'ensemble énuméré $\tilde{\mathbb{I}}$ par les paramètres p, q, r, λ. Chacun de ces paramètres, en résumé, correspond à :

1. k_{max} : le nombre maximum de segments de vol que peut comporter un itinéraire ;
2. t_{min} : le temps minimum requis pour assurer au passager la possibilité de pouvoir

réaliser une correspondance ;

3. t_{max} : le temps maximal qu'un passager est disposé à attendre entre deux segments de vol d'un itinéraire ;
4. λ : le rapport entre le temps requis pour réaliser l'itinéraire par rapport au meilleur temps qu'il est possible d'obtenir entre les mêmes origine et destination, à tout moment de la semaine.

Ces quatre paramètres doivent être fixés avant de procéder à l'énumération. Nous pourrons cependant penser à une manière d'évaluer $T_{INF}(\mathcal{O}(I), \mathcal{D}(I))$ dynamiquement.

3.4 Ré-optimisations locales

Le processus de génération d'itinéraires de passagers est appelé à être exécuté lors d'itérations multiples tel que présenté dans le chapitre 1. Lorsque nous procédons à une première itération, numérotée 0, nous obtenons l'ensemble $\tilde{\mathbb{I}} = \mathbb{I}_0$. Une itération survient donc lorsqu'au moins un changement à l'horaire des vols survient par rapport à ce qui était établi en tout premier lieu. L'énumération amenant à l'obtention de l'ensemble $\mathbb{I}_1$ devra s'effectuer sur un réseau espace-temps légèrement différent de celui à l'itération précédente. Ceci est vrai pour une k-ième itération, lorsque $k \geq 1$. Mais la nature itérative du processus d'améliorations successives fait en sorte que les modifications sont seulement partielles ou locales, c'est-à-dire que les arcs touchés par des modifications, ajouts et suppressions confondues, se retrouvent groupés dans une ou quelques régions du réseau. En effet, un changement d'un type d'avion peut avoir une incidence sur le temps total d'un segment de vol, ce qui aura inévitablement un effet sur soit l'évènement de décollage ou l'évènement d'atterrissage. On a vu précédemment qu'un segment de vol S se traduit par un arc $\tilde{S}$ dans le réseau.

Il faudra pouvoir considérer ces modifications mineures dans le réseau lors de ré-optimisations locales afin de réduire le temps de calcul dans le contexte où un processus itératif avec plusieurs boucles avec des changements de types d'avion ou des annulations de vols pourront avoir lieu.

Ainsi donc, nous avons déjà identifié par $\mathbb{S}$ l'ensemble de tous les segments de vol définis pour un transporteur aérien. Considérons l'ensemble initial de ces segments de vol par $\mathbb{S}_0$. Au moment de débuter l'itération suivante nécessaire pour tenir compte

de modifications de types d'avion et d'annulations de vols, on aura un ensemble $\mathbb{S}_1$ représentant un nouvel ensemble de segments de vol disponibles au début de la première itération. Directement, on obtiendra l'ensemble $\mathbb{S}' = \mathbb{S}_1 \setminus \mathbb{S}_0$ de segments de vol ayant été modifiés depuis l'itération précédente. Conséquemment, il faudra retirer les itinéraires faisant utilisation des segments de vol $s \in \mathbb{S}'$.

Définissons une fonction qui nous donne l'ensemble I des itinéraires comportant le segment de vol s :

$$\phi(s, \mathbb{I}) = \{\ I \mid s \in I, I \in \mathbb{I}\ \} \tag{3.5}$$

On peut considérer que $\phi(s, \mathbb{I})$ représente un sous-ensemble des itinéraires $\mathbb{I}$ qui dépendent de s, ou dans lesquels le segment de vol s est emprunté. Par extension, on peut aussi noter l'ensemble de tous les itinéraires affectés par tous les vols ayant été modifiés ou annulés. Utilisons la notation suivante pour décrire les segments de vol ayant été annulés entre l'itération $j-1$ et l'itération j par $\mathbb{S}'_j = \mathbb{S}_{j-1} \setminus \mathbb{S}_j$.

$$\phi(\mathbb{S}', \mathbb{I}) = \bigcup_{s \in \mathbb{S}'} \{\ I \mid s \in I,\ I\ \in \mathbb{I}\} \tag{3.6}$$

Pour énumérer l'ensemble des itinéraires $\mathbb{I}_j$ qui sont valables pour le nouvel horaire de vols ayant donné lieu à $\mathbb{S}_j$, on procédera comme suit :

1. Identifier les arcs (segments de vol) qui étaient présents dans l'itération $j-1$ mais qui ont été annulés et ne sont plus présents dans l'itération j. Cet ensemble est $\mathbb{S}'_j = \mathbb{S}_{j-1} \setminus \mathbb{S}_j$;
2. Déterminer la portée des annulations de segments de vol en termes d'itinéraires invalidés par l'annulation de l'un ou l'autre des segments de vol ayant été annulés ;
3. Déterminer la portée des annulations de segments de vol en termes de paires d'aéroports (O, D) où O représente un aéroport d'origine et D un aéroport de destination.

Chapitre 4

ALGORITHMES

Ce chapitre présente deux algorithmes développés dans le cadre des présents travaux de recherche. Chaque algorithme a pour but de procéder au dénombrement de l'ensemble d'itinéraires de passagers voulant voyager entre toute paire d'aéroports pour lesquelles on connaît une demande de transport ; c'est l'ensemble $\mathbb{I}$ présenté à la section 3.3. Les algorithmes présentés sont de natures différentes. Le premier algorithme effectue une recherche en profondeur dans un réseau espace-temps en faisant usage d'une heuristique permettant d'appliquer une technique d'accélération par émondage de l'arbre de recherche. Le deuxième algorithme, quant à lui, est de nature combinatoire et ne requiert aucun graphe ou réseau espace-temps. Dans chacun des cas, les algorithmes font appel à des structures de données qui sont fondamentales au bon déroulement des algorithmes et à leur performance ; elles seront présentées au fur et à mesure que leur nécessité se présentera.

Nous commençons ce chapitre par la présentation de l'heuristique qui sera utilisée dans le premier algorithme. Nous poursuivons avec la présentation des deux algorithmes d'énumération des itinéraires, chacun dans sa forme initiale requérant un pré-traitement nécessaire pour renseigner l'heuristique et ensuite dans sa version dynamique, c'est-à-dire une variante de chacun des algorithmes où les informations requises par l'heuristique sont mises à jour dynamiquement par un processus de relaxation des valeurs numériques ; on cherche donc à pouvoir éliminer les pré-calculs dans les situations où cela s'avèrerait utile. Pour la méthode combinatoire, ceci a pour conséquence d'éliminer complètement la nécessité d'avoir un réseau espace-temps tel que présenté dans la section 3.2 ; seules les données sur les segments de vol seront alors requises de pair avec des structures de données auxiliaires pour retrouver certaines informations rapidement.

Nous verrons finalement une méthode permettant d'opérer des optimisations locales avec l'algorithme combinatoire après qu'une perturbation à l'horaire de vols ait eu lieu.

4.1 Heuristique

Le domaine de l'intelligence artificielle, fort de la théorie des graphes, a permis l'éclosion de plusieurs algorithmes permettant de trouver un ou plusieurs chemins dans un graphe le plus rapidement possible pour tant peu qu'on puisse modéliser un problème sous forme de graphe. L'algorithme A* est l'un de ceux-ci (Dechter et Pearl (1985), Russell et Norvig (1995), Koenig *et al.* (2004)) et utilise une heuristique pour accélérer la recherche. Dans sa forme originale, c'est une méthode de type *best-first* qui s'arrête dès lors qu'une première solution optimale est trouvée. Dans le cas de A*, l'heuristique a pour but d'estimer le coût final d'une solution S en cours de confection et d'éviter des recherches inutiles ; elle permet d'émonder l'arbre de recherche. Pour A*, une solution est un chemin dans un graphe ou un réseau. En effet, si nous souhaitons évaluer le coût d'une solution S, ou d'un itinéraire de passagers dans notre cas, il serait utile de pouvoir l'évaluer à l'aide d'une fonction

$$f^*(S) \approx f(S) = g(S) + h(S) \tag{4.1}$$

où :

- f^* est la fonction donnant le coût réel d'une solution S ;
- f est la fonction visant à évaluer le coût final d'une solution S, $f(S)$ donne une borne inférieure de la valeur de $f^*(S)$;
- g est la valeur du coût pour la portion connue de S ;
- h est la fonction heuristique qui estime le coût de la portion inconnue de la solution S.

Dans le cas qui nous concerne, nous avons qu'une solution est un itinéraire en cours de confection.

La définition 4.1.1 exprime une condition que peut satisfaire certaines fonctions heuristiques à utiliser avec l'algorithme A*.

Definition 4.1.1 (Russell et Norvig (1995))**.** Une fonction heuristique h qui évalue le coût d'une solution S partant d'un sommet initial s vers un sommet terminal t est dite *monotone* (ou *consistante*) si et seulement si pour chaque sommet n et chaque descendant m de n, le coût estimé pour atteindre t à partir de n est inférieure ou égal à la somme du coût pour aller de n vers m et du coût estimé pour se rendre de m vers le sommet t. Formellement, si $G = (V, E, \omega)$ est un réseau, ou un graphe orienté

pondéré, alors

$$h(n,t) \leq c(n,m) + h(m,t),\ \forall n,t \in V,\ \forall m \in V \mid (n,m) \in E \tag{4.2}$$

pour tout successeur m de n où n est un sommet rencontré dans l'élaboration de S.

La définition 4.1.1 correspond en fait à l'inégalité triangulaire exprimée dans le cas précis d'un chemin dans un graphe modélisant un problème.

Le fait d'utiliser une heuristique monotone est d'autant plus intéressant que cela garantit que l'algorithme A* est *complet* lorsqu'il s'agit d'une recherche dans un graphe quelconque, c'est-à-dire qu'une solution sera trouvée, si une telle solution existe, et que cette solution est optimale.

Remarquons aussi que l'algorithme de Dijkstra est un cas particulier de A* dont l'heuristique h_D pour l'algorithme de Dijkstra est donnée par $h_D = h(x) = 0, \forall x$, ce qui a pour effet d'augmenter la taille de l'espace de recherche. Cette heuristique est évidemment optimiste, quoique simpliste, mais donne de bons résultats. On a de manière triviale que $h_D(x)$ est monotone, garantie alors que l'algorithme de Dijkstra est complet, c'est-à-dire qu'il trouvera une solution optimale si elle existe.

En prenant pour h la borne inférieure sur le temps de parcours de l'itinéraire restant, nous sommes alors certain que h soit monotone croissante. Et nous pouvons même être certain que h est monotone strictement croissante car aucun arc du graphe n'est de poids nul ou négatif. Nous sommes ainsi assurés de respecter l'inégalité (4.2). Nous devons donc avoir une borne inférieure sur le temps de parcours entre chaque paire d'aéroports. La distance euclidienne entre deux points est souvent utilisée comme heuristique dans des cas où la recherche correspond à trouver un chemin ou un parcours dans l'espace en trois dimensions, par exemple. Mais nous sommes ici intéressés par les temps de transport et les détours géographiques ne sont d'aucun intérêt. Voilà donc pourquoi la distance géographique ne sied pas.

Selon Pearl et Korf (1987), les risques associés à l'utilisation de l'algorithme A* concernent surtout la capacité mémoire. Mais ce trait est commun à toute variante de la méthode *best-first* car en tout temps, les amorces des solutions doivent être conservées en mémoire. D'autres algorithmes ont été développés afin de contourner cette limitation le cas échéant. Mais nous n'avons observé aucun problème avec le jeu de données sur une machine qui soit en fait une station de travail de bureau ; plus de détails au sujet de la machine cible sont fournis à la section 5.4.

Definition 4.1.2 (Luger et Stubblefield (1998)). Si nous avons $h_1(S)$ et $h_2(S)$, deux heuristiques pour une estimation du coût de la solution S, nous dirons que h_2 est *mieux informée* que h_1 si

$$h_1(S) < h_2(S) \leq C(S),\ \forall S \tag{4.3}$$

où $C(S)$ représente le coût réel de S.

Ainsi, avec une heuristique admissible qui soit non-triviale, comme c'est le cas avec l'heuristique utilisée par l'algorithme de Dijkstra, nous avons que l'espace de recherche est diminué avec une heuristique qui soit mieux informée car

$$0 = h_D(S) < h(S),\ \forall S$$

où $h(S)$ est une heuristique non-triviale telle que déterminée par les temps minimums de voyage par exemple entre deux aéroports.

4.1.1 Pré-traitement

Afin de pouvoir obtenir une heuristique optimiste $h(S)$ sous la main et être en mesure d'évaluer le temps restant à un itinéraire I en cours d'élaboration, nous devons calculer *a priori* le plus court temps de déplacement qu'il est possible d'effectuer entre chaque paire d'aéroports. Nous exécuterons un pré-traitement qui donnera cette information. Le calcul de la borne inférieure $T_{min}(O, D)$ sur le temps de voyage entre chaque paire d'aéroports s'effectue avec une implémentation de l'algorithme de Dijkstra ; cet algorithme s'exécutera sans problème sur un réseau avec cycles, ce qui est le cas de notre réseau qui représente un horaire cyclique sur un horizon d'une semaine. Nous nous limiterons à calculer cette heuristique pour les paires (O, D) faisant l'objet d'une demande de transport non-nulle.

Nous pourrons ensuite utiliser $T_{min}(O, D)$ à titre d'heuristique $h(S)$ car D est l'aéroport de destination qu'on souhaite atteindre et tout sommet associé à D sera considéré comme sommet terminal, c'est-à-dire l'objectif de la solution S, et O pourra être n'importe quel aéroport intermédiaire.

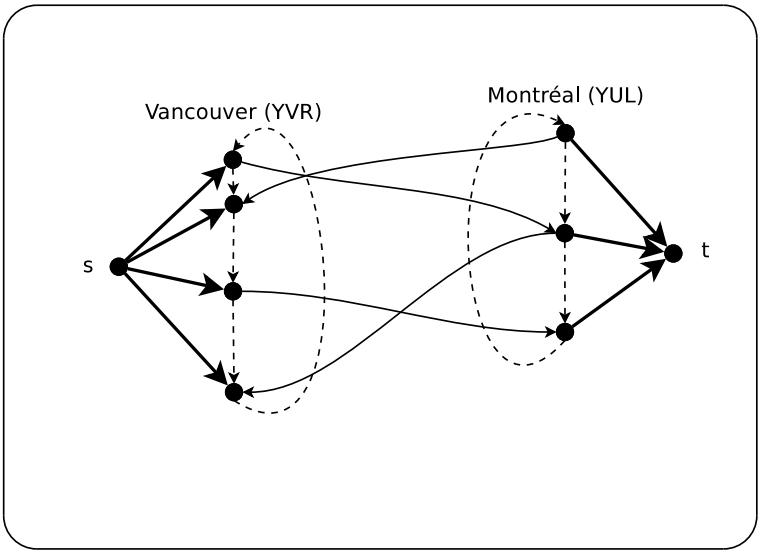

FIGURE 4.1 Sommets source s et puits t pour le pré-traitement

L'algorithme du plus court chemin trouve une solution pour une paire de sommets. Pour une paire d'aéroports donnée, le plus court chemin peut être constitué de plus d'un segment dans le cas où aucun vol direct ne relie ces aéroports. Mais nous cherchons à calculer le plus court chemin entre deux aéroports qui correspondent en fait chacun à un ensemble de sommets distincts. Il suffira d'ajouter un sommet artificiel s qui représentera un sommet source rattaché par un arc (s, u) de poids nul à tous les sommets u associés à l'aéroport d'origine. De manière analogue, on ajoutera un sommet artificiel t représentant un puits et autant d'arcs (v, t) de poids nul que de sommets v associés à l'aéroport de destination. La figure 4.1 présente visuellement cette stratégie. Calculer le plus court temps entre s et t suffira pour l'obtention de l'information désirée. Après coup, on retire les sommets s et t ainsi que tous les arcs rattachés à ces sommets et arcs factices et on reprend pour une autre paire d'aéroports. L'algorithme 4.1.1 décrit cette procédure.

Remarque 2. Nous considérons que $T_{min}(O, D) = \infty, \ \forall \ (O, D) \ \in \ \mathbb{A}^2 \setminus \mathbb{A}^2_*$.

Remarque 3. Les valeurs de bornes inférieures de temps de voyage calculées (T_{min}) sont emmagasinées dans une table de hachage pour permettre un accès rapide lors de consultations futures.

Notons que si les informations sur les bornes de temps sont disponibles *a priori*,

Algorithme 4.1.1 INITIALISER(T_{min}, $\mathbb{A}^2_*$), G

ENTRÉE: $\mathbb{A}^2_* = \{ (O,D) \mid \Delta(O,D) > 0 \}$: l'ensemble des paires d'aéroports pour lesquels une demande existe
ENTRÉE: $G = (V, E, \omega)$: un réseau.
SORTIE: Les valeurs de la fonction $T_{min}(O,D)$ sont calculées, $\forall (O,D) \in \mathbb{A}^2_*$.

1: **POUR TOUT** $(O,D) \in \mathbb{A}^2_*$ **FAIRE**
2: $V' \leftarrow V \cup \{s,t\}$
3: $E' \leftarrow E$
4: $\omega' \leftarrow \omega$ // une copie de la fonction $\omega : \mathbb{A}^2 \rightarrow \mathbb{R}$
5: **POUR TOUT** sommet $u \in V$ associé à l'aéroport O **FAIRE**
6: $E' \leftarrow E' \cup \{(s,u)\}$
7: $\omega'((s,u)) \leftarrow 0$
8: **POUR TOUT** sommet $v \in V$ associé à l'aéroport D **FAIRE**
9: $E' \leftarrow E' \cup \{(v,t)\}$
10: $\omega'((v,t)) \leftarrow 0$
11: $p \leftarrow Dijkstra(s,t,(E',V',\omega'))$ // p est plus court chemin de s vers t
12: $T_{min}(O,D) \leftarrow$ temps du chemin p
13: **RETOURNER** T_{min}

alors le pré-traitement peut être éliminé. Ces informations pourraient aussi être dictées par des contraintes de marché. Mais comme nous considérons que ce n'est pas toujours le cas, un pré-traitement règle la question.

4.2 Recherche en profondeur

Le premier algorithme consiste en une recherche en profondeur dans le graphe espace-temps mais qui utilise une heuristique pour réduire l'espace de recherche, comme c'est le cas avec l'algorithme A*. L'heuristique utilisée est celle décrite précédemment. L'adaptation de cet algorithme vers une méthode de recherche des itinéraires de passagers intéressants dans notre réseau se traduit d'abord par la limitation de la profondeur de recherche à un nombre constant de segments de vol qui est fixé *a priori* ; les arcs d'attente ne sont pas comptabilisés dans le calcul de profondeur de recherche. Ensuite, on effectue du retour sur trace, ou *backtracking*, lorsque l'une ou l'autre des contraintes déterminées n'est pas satisfaite par l'itinéraire en cours de confection. Pour une paire d'aéroports (O,D) faisant l'objet d'une demande de transport, on peut lancer la procédure successivement sur chacun des sommets rattachés à cet aéroport et ainsi trouver tous les itinéraires permettant de rejoindre l'aéroport

D donné initialement. De manière naturelle, l'algorithme de recherche en profondeur se définit de manière récursive.

On procède avec un étiquetage des sommets du réseau $G = (V, E, \omega)$. Les informations d'une étiquette d'un sommet $v \in V$ sont accessibles de cette manière :

(i) $v.etiquette.cout$: coût cumulatif pour atteindre le sommet v ;

(ii) $v.etiquette.nbSegments$: nombre de segments de vol rencontrés depuis l'origine jusqu'à v ;

(iii) $v.etiquette.attente$: temps total d'attente à l'aéroport associé à v ;

(iv) $v.etiquette.precedent$: sommet précédent à v ;

(v) $v.etiquette.arc$: arc emprunté vers v.

Une seule étiquette est rattaché à chaque sommet du réseau car nous utilisons un algorithme récursif ; un seul itinéraire est en cours de confection à chaque instant.

L'algorithme 4.2.1 sert à amorcer une procédure de recherche récursive à la ligne 11 pour chacun des sommets de l'aéroport de départ O. L'algorithme récursif est décrit par l'algorithme 4.2.2. La ligne 11 de l'algorithme 4.2.2 gère le cas où un nouvel itinéraire est trouvé ; l'ensemble I est mis à jour avec un nouvel item.

L'algorithme 4.2.3 montre de quelle manière les étiquettes de chaque sommet sont mises à jour avant un appel récursif subséquent. Notons que la fonction $depart(e)$ retourne le sommet initial de l'arc e et la fonction $arrivee(e)$ retourne le sommet final de l'arc e. La ligne 5 vérifie s'il s'agit d'un arc explicite. Si c'est le cas, alors on considère que la profondeur de recherche est augmentée d'une unité. Sinon, le compte demeure inchangé.

Chaque étiquette associée à un sommet est une propriété qui est accessible dans un temps constant. Il en va de même pour les champs contenus dans chaque étiquette. On peut donc obtenir un temps d'accès à ces informations qui soit rapide.

4.2.1 Version dynamique

La section 4.1.1 a présenté le pré-traitement qui permet d'obtenir des bornes intérieures sur les temps de voyage entre chaque paire d'aéroports (O, D) et ainsi avoir une heuristique pour accélérer les temps de calculs. Mais le pré-traitement requis afin de calculer T_{min} pour chaque paire d'aéroports (O, D) avec un algorithme de plus court chemin requiert un certain temps d'exécution avant même de lancer un

Algorithme 4.2.1 RECHERCHE-EN-PROFONDEUR(O, D, G, T_{min}, t_{min}, t_{max}, k_{max})

ENTRÉE: O : l'aéroport d'origine
ENTRÉE: D : l'aéroport de destination
ENTRÉE: $G = (V, E, \omega)$: un réseau
ENTRÉE: T_{min} : les bornes inférieures de temps de voyage
ENTRÉE: t_{min} : le temps minimum d'attente dans un aéroport (contrainte)
ENTRÉE: t_{max} : le temps maximum d'attente dans un aéroport (contrainte)
ENTRÉE: k_{max} : le nombre maximum de segments de vol permis (contrainte)
ENTRÉE: λ : le rapport maximal du temps d'itinéraire entre O et D par rapport au meilleur temps pour O et D
SORTIE: L'ensemble I des itinéraires satisfaisant les contraintes

```
1:  réinitialiser les étiquettes des sommets de G
2:  INITIALISER(T_min)
3:  I ← ∅
4:  POUR TOUT sommet σ associé à O représentant un décollage FAIRE
5:      réinitialiser les étiquettes de σ
6:      coût cumulatif ← 0
7:      temps d'attente ← 0
8:      sommet précédent ← ∅
9:      arc emprunté pour se rendre à ce sommet ← ∅
10:     nombre de segments rencontrés ← 0
11:     RECHERCHE-RÉCURSIVE( G, σ, D, T_min, t_min, t_max, k_max, λ, I)
12: RETOURNER I
```

calcul d'énumération d'itinéraires de passagers. Des modifications à l'horaire de vols impliquent des modifications au réseau et possiblement des changements aux valeurs de bornes inférieures. De plus, dans le contexte global d'un système d'optimisation de lignes aériennes, des modifications à l'affectation de flottes pourraient avoir eu lieu et modifier les valeurs de $T_{min}(O, D)$, ou $h(S)$, même si l'horaire est resté inchangé. Recalculer les bornes inférieures à chaque itération représente alors un coût qui paraît prohibitif. On voit donc aisément un avantage à réduire le temps de calcul de ces bornes inférieures, quitte à utiliser seulement une estimation de ces valeurs.

Considérons $T'_{min}(O, D)$, une approximation de $T_{min}(O, D)$ mise à jour dynamiquement. Nous désignons une valeur par défaut pour chaque paire (O, D) qui soit excessivement grande initialement et graduellement ajustée ou relaxée vers la valeur la plus petite au fur et à mesure que des itinéraires sont générés. Rappelons que T'_{min} représente les valeurs de bornes inférieures de temps de voyage entre chaque paire

Algorithme 4.2.2 RECHERCHE-RÉCURSIVE(G, σ, D, T_{min}, t_{min}, t_{max}, k_{max}, λ, I)

ENTRÉE: $G = (V, E, \omega)$: un réseau
ENTRÉE: σ : un sommet de G ($\sigma \in V$)
ENTRÉE: D : l'aéroport de destination
ENTRÉE: T_{min} : l'information sur les bornes inférieures de temps
ENTRÉE: t_{min} : le temps minimum d'attente admis dans un aéroport
ENTRÉE: t_{max} : le temps maximum d'attente admis dans un aéroport
ENTRÉE: k_{max} : le nombre maximum permis de segments de vol
ENTRÉE: λ : le rapport maximal du temps d'itinéraire entre O et D par rapport au meilleur temps pour O et D
ENTRÉE: I : l'ensemble solution pour mise à jour

```
1:  N ← σ.etiquette.nbSegments // nb de segments empruntés jusqu'à ce sommet
2:  C ← σ.etiquette.cout // coût cumulatif jusqu'à ce sommet
3:  A ← σ.aeroport // aéroport actuel
4:  W ← σ.etiquette.attente // temps d'attente
5:  R ← T_min(A, D) // temps estimé pour le reste de l'itinéraire, l'heuristique
6:  P ← C + R // temps estimé pour l'itinéraire en construction
7:  μ ← λ × T_min(O, D)
8:  SI (W > t_max) ∨ (W ≤ t_min) ∨ (N > k_max) ∨ (P > μ) ALORS
9:     TERMINER
10: SI A = D ALORS
11:    i : un nouvel l'itinéraire menant au sommet courant
12:    I ← I ∪ {i} : l'ensemble des itinéraires est mis à jour
13: POUR TOUT v ∈ { p | (σ, p) ∈ E} FAIRE
14:    ÉTIQUETER(G, v, (σ, v))
15:    RECHERCHE-RÉCURSIVE(G, v, D, T_min, t_min, t_max, k_max, λ, I)
```

d'aéroports. Les temps sont exprimés en minutes et on compte au plus 10880 minutes dans une semaine. Une très grande valeur initiale de T'_{min} pour une paire (O, D) donné nous amènera à conserver des itinéraires de passagers qui ne seraient pas conservés avec une heuristique mieux renseignée mais, en procédant avec une relaxation des valeurs de $T'_{min}(O, D)$ on pourra réduire graduellement le nombre d'itinéraires de mauvaise qualité qui sont conservés.

Cette approche ressemble, à certains égards, à l'algorithme de Dijkstra qui considère un temps d'accès vers un sommet comme étant arbitrairement grand et le réduit successivement à chaque fois qu'une nouvelle valeur, meilleure que la précédente, est trouvée. Nous espérons ainsi atteindre un compromis entre la qualité de l'ensemble d'itinéraires énumérés et le temps de calcul. Il est clair que l'heuristique n'est plus optimiste, ce qui implique qu'on pourrait augmenter l'espace de recherche à explorer par

Algorithme 4.2.3 ÉTIQUETER(G, v, e)

ENTRÉE: $G = (V, E, \omega)$: un réseau
ENTRÉE: $v \in V$: un sommet du réseau
ENTRÉE: $e = (u, v) \in E$: un arc du réseau avec $u \in V$
SORTIE: Les valeurs contenues dans l'étiquette du sommet s sont mises à jour

1: $u \leftarrow depart(e)$
2: $v.etiquette.arc \leftarrow e$
3: $v.etiquette.precedent \leftarrow u$
4: $v.etiquette.cout \leftarrow u.etiquette.cout + \omega(e)$
5: **SI** $v.etiquette.aeroport \neq u.etiquette.aeroport)$ **ALORS**
6: $v.etiquette.nbSegments \leftarrow u.etiquette.nbSegments + 1$
7: $v.etiquette.attente \leftarrow 0$
8: **SINON**
9: $v.etiquette.nbSegments \leftarrow u.etiquette.nbSegments$
10: $v.etiquette.attente \leftarrow u.etiquette.attente + \omega(e)$

l'algorithme. Nous devrons évaluer numériquement les impacts d'une telle stratégie.

L'algorithme 4.2.2 peut donc facilement être modifié afin d'utiliser la version avec mise à jour dynamique des bornes inférieures des temps de voyage. Il faut mettre à jour la structure de données T'_{min} après qu'un nouvel itinéraire est trouvé dans l'algorithme 4.2.2, soit à la ligne 11.

4.2.2 Complexité temporelle

Au sens le plus strict, une recherche en profondeur dans un graphe $G = (V, E)$ où $|V| = n$ et $|E| = m$ nécessite qu'on emprunte au maximum m chemins à partir de chacun des n sommets. Nous avons donc une complexité temporelle de $O(n \times m)$ car tous les sommets et toutes les arêtes, ou arcs dans le cas d'un graphe orienté, sont visitées. Dans notre cas plus particulier, nous parcourons le graphe en profondeur mais sur une profondeur limite, disons k_{max} arcs. De plus, si le degré sortant moyen d'un sommet est d et que nous nous limitions à une profondeur de recherche de k_{max} arcs, alors nous avons que la complexité temporelle est $O(n \times d^{k_{max}})$. Mais les arcs d'attente sont négligés dans le calcul de la profondeur de la recherche car seul leur coût, en minutes, est comptabilisé pour la période d'attente dans un même aéroport. Pour chacune des paires (O, D), nous aurons alors que $n^2 \times d^{k_{max}}$ est une borne inférieure et ainsi la complexité devrait plutôt être $O(n^2 \times d^{k_{max}})$.

Mais il serait utile de pouvoir exprimer la complexité temporelle en fonction du nombre de sommets n et d'arcs m de notre réseau. Le *lemme des coups de pieds* (Labelle (1981)) nous dit ceci :

Lemme 4.2.1. *Pour tout graphe orienté $G = (V, A)$, nous avons que*

$$\sum_{v \in V} d^+(v) = \sum_{v \in V} d^-(v) = |A|$$

où

(i) $d^+(v)$ *est le degré sortant de* v *;*

(ii) $d^-(v)$ *est le degré entrant de* v *;*

(iii) $|A| = m$.

Or, le degré sortant moyen est donné par ceci :

$$d = \frac{1}{n} \sum_{\forall v \in V} d^-(v) = \frac{1}{n} \sum_{\forall v \in V} d^+(v) = \frac{m}{n}$$

La profondeur de recherche sera paramétrisée par le terme k_{max}. Donc, la complexité temporelle est $O\left(n^2 \times \left(\frac{m}{n}\right)^{k_{max}}\right)$. Et le temps de calcul est exponentiel par rapport au nombre maximum de segments de vols permis dans un itinéraire.

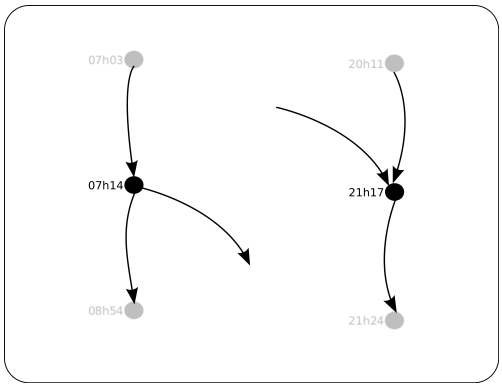

FIGURE 4.2 Décollage et atterrissage dans le réseau

Or, nous avons un réseau où chaque sommet est dans l'une ou l'autre des situations présentées à la figure 4.2. On peut donc raffiner notre approximation à ceci :

$$O\left(n^2 \times \left(\frac{3}{2}\right)^{k_{max}}\right)$$

La complexité des algorithmes présentés ci-haut est grandement affectée par la complexité des structures de données utilisées pour accéder aux données. Entre autres, la structure de données $T_{min}(O, D)$ est consultée à chaque visite d'un sommet. C'est pourquoi l'implémentation de cette structure de données sous forme de table de hachage semble un choix judicieux : accéder à une donnée est constant, ou $O(1)$.

4.3 Méthode combinatoire

Contrairement à la méthode décrite dans la section 4.2, nous considérons maintenant une méthode de nature combinatoire. Mettons de côté le paradigme qui fait appel au modèle mathématique, le réseau, et voyons comment aborder le problème d'une autre manière. En effet, la méthode qui sera présentée ci-après ne fait appel à aucun modèle mathématique particulier, sinon pour effectuer le pré-calcul des bornes inférieures. Mais un tel réseau n'est plus requis dès lors que l'information sur les bornes de temps est disponible autrement. Et une technique de relaxation de $T'_{min}(O, D)$ comme celle décrite précédemment permet même d'éviter de devoir effectuer un pré-traitement et ainsi épargner la création d'un réseau espace-temps tout en fournissant une évaluation sur les bornes de temps de transport.

Pour énumérer des combinaisons d'éléments dans un ensemble, il est naturel de considérer des combinaisons. Déterminer les combinaisons de p éléments qu'il est possible d'obtenir à partir d'un ensemble de n éléments représente un nombre extravagant d'items, à savoir $\binom{n}{p} = \frac{n!}{p!(n-p)!}$. Il faut donc éviter la méthode d'énumération exhaustive dont le coût est prohibitif.

La présente méthode procède de manière itérative sur le nombre de segments de vol qui constituent un itinéraire. On utilise ici une définition récursive d'un itinéraire de passagers, à savoir :

Definition 4.3.1. Un *itinéraire de passagers* dans un réseau de transport aérien est soit

1. constitué d'un unique segment de vol $s \in \mathbb{S}$;
2. constitué d'un itinéraire i suivi d'un segment s.

À partir de cette définition, si nous avons que l'itinéraire i_s est constitué du seul segment s, alors $\mathcal{L}(i_s) = 1$ dans le premier cas. Dans le second cas, si nous avons que $\mathcal{L}(i) = n$ et que $i' = i + i_s$, alors $\mathcal{L}(i') = \mathcal{L}(i) + \mathcal{L}(i_s) = n + 1$.

Bien entendu, au moment de construire un nouvel itinéraire i', il faut satisfaire $\Pi(i', k_{max}, t_{min}, t_{max}, \lambda) > 0$ tel que décrit à l'équation (3.3) afin que i' puisse être considéré dans l'ensemble $\mathbb{I}$ défini à l'équation (3.4). Les deux points de la définition 4.3.1 constituent notre algorithme initial.

Afin de réduire l'espace de recherche, on cherche à réduire les combinaisons à évaluer par la fonction Π sur le candidat $i' = i_L + i_R$. Par exemple, on voudra ne considérer que les itinéraires initiaux i_L dont la destination, disons A, coïncide avec l'origine de l'itinéraire terminal i_R. Aussi, si on souhaite produire les itinéraires dont le nombre de segments de vol est limité à k_{max} segments de vol, alors on souhaitera obtenir d'une part les itinéraires constitués de $k_{max} - 1$ segments et atterrissant à l'aéroport A et tenter de les combiner avec seulement les itinéraires à un seul segment de vols qui décollent de l'aéroport A.

Dans chaque cas, on doit retrouver rapidement des sous-ensembles d'itinéraires selon le nombre de segments de vol qu'ils comportent et aussi selon un aéroport d'origine ou de destination. Ces sous-ensembles peuvent être décrits par des conteneurs séquentiels tels que des listes ou des tableaux. Mais la recherche dans de telles structures de données doit pouvoir s'effectuer rapidement. Un choix judicieux de structures de données doit donc être fait afin de réduire les temps d'accès aux données pertinentes. Commençons par décrire quelques unes des structures de données utilisées dans la méthode combinatoire. Nous avons :

- des listes telles que des listes simplement chaînées ;
- des tables de hachage ;
- des arbres binaires de recherche tels que les arbres rouge-noir ou AVL (Adelson-Velskii et Landis (1962)).

Pour accéder aux sous-ensembles pertinents, nous utiliserons ici les tables de hachage et arbres binaires de recherche à titre de conteneurs associatifs, c'est-à-dire que nous pouvons stocker ou retrouver une valeur dans la structure de données à l'aide d'une clé de recherche. La complexité temporelle d'accès à un item dans une table de hachage contenant n éléments est constante, soit de complexité temporelle $O(1)$, alors que pour un arbre binaire de recherche comportant n items, cette complexité est plutôt $O(\log(n))$.

Nous avons plus particulièrement :

- un ensemble mathématique implémenté par une table de hachage de tous les aéroports pour lesquels au moins un départ, ou décollage, survient ;
- un ensemble mathématique implémenté par une table de hachage de tous les aéroports pour lesquels au moins une arrivée, ou atterrissage, survient ;
- une table de hachage qui donne, pour un aéroport donné, un arbre binaire de recherche permettant d'obtenir une liste d'itinéraires décollant à cet aéroport qui sont composés d'un nombre de segments de vol donné ;
- une table de hachage qui donne, pour un aéroport donné, un arbre binaire de recherche permettant d'obtenir une liste d'itinéraires atterrissant à cet aéroport qui sont composés d'un nombre de segments de vol donné.

Ces structures de données permettent donc aisément de trouver les sous-ensembles pertinents d'itinéraires qui sont susceptibles d'être impliqués dans une combinaison. La recherche peut s'effectuer pour un lieu d'origine ou de destination et aussi selon le nombre de segments de vol qui constituent les itinéraires.

4.3.1 Pseudo-code

L'algorithme 4.3.1 présente le pseudo-code décrivant la méthode de recherche combinatoire. Les lignes 4 et 7 de cet algorithme servent respectivement à mettre à jour les structures de données décrites ci-haut pour chaque nouvel itinéraire, ici composés d'un seul segment de vol, et procéder ensuite à la génération de connexions avec ces itinéraires. La ligne 4 de l'algorithme 4.3.1 est explicitée par l'algorithme 4.3.2.

Dans l'algorithme 4.3.2, la ligne 7 procède en insérant dans un conteneur associatif, une table de hachage en l'occurrence, un arbre binaire de recherche dont la clé de recherche est le nom de l'aéroport où l'itinéraire i se termine. L'arbre binaire de recherche est, lui aussi, un conteneur associatif où nous avons que la clé de recherche est un entier, à savoir le nombre de segments de vol qui constituent les itinéraires arrivant à l'aéroport A. On permet donc de circonscrire le domaine de recherche à un sous-ensemble I' d'itinéraires de passagers en fonction de 1) l'aéroport d'arrivée et 2) le nombre de segments de vol qui constituent les itinéraires visés. La ligne 6 procède de manière tout à fait analogue mais ce sont les aéroports de départ qui sont utilisés comme première clé de recherche. Nous avons donc deux structures de données de même nature mais utilisées à des fins distinctes.

Algorithme 4.3.1 RECHERCHE-COMBINATOIRE(T_{min}, t_{min}, t_{max}, $\mathbb{S}$, λ, k_{max})

ENTRÉE: T_{min} : l'information sur les bornes inférieures de temps
ENTRÉE: t_{min} : le temps minimum pour une connexion dans un aéroport
ENTRÉE: t_{max} : le temps maximum d'attente dans un aéroport
ENTRÉE: $\mathbb{S}$: l'ensemble des segments de vol
ENTRÉE: λ : le rapport maximal du temps d'un itinéraire entre O et D par rapport au meilleur temps de voyage de O vers D
ENTRÉE: k_{max} : le nombre maximal de segments de vol par itinéraire
SORTIE: I : l'ensemble des itinéraires générés
1: $I \leftarrow \emptyset$
2: **POUR TOUT** $s \in \mathbb{S}$ **FAIRE**
3: i est un itinéraire composé uniquement de s
4: CONNAÎTRE(i)
5: $I \leftarrow I \cup \{i\}$
6: **POUR TOUT** $p \in \{2, \ldots, k_{max}\}$ **FAIRE**
7: CONNECTER($p-1$, 1, t_{min}, t_{max}, T_{min}, λ, I)
8: **RETOURNER** I l'ensemble solution

Algorithme 4.3.2 CONNAÎTRE(i)

ENTRÉE: i : itinéraire de passagers
1: $A \leftarrow \mathcal{O}(i)$: l'origine de i
2: $B \leftarrow \mathcal{D}(i)$: la destination de i
3: $p \leftarrow \mathcal{L}(i)$: le nombre de segments de vol de i
4: $O \leftarrow O \cup \{A\}$
5: $D \leftarrow D \cup \{B\}$
6: $L^+(A,p) \leftarrow L^+(A,p) + i$ // mettre à jour la structure de données des départs
7: $L^-(B,p) \leftarrow L^-(B,p) + i$ // mettre à jour la structure de données des arrivées

Par exemple, si notre structure de données pour accéder à la liste des itinéraires qui décollent de l'aéroport A et qui sont constitués de v segments de vol est identifiée par L^+, on pourra identifier la liste d'itinéraires correspondante par $L^+(A,t)$. De manière analogue, nous aurons $L^-(B,u)$ pour la liste des itinéraires qui atterrissent à l'aéroport B et composés de w segments de vol.

Algorithme 4.3.3 CONNECTER(v, w, t_{min}, t_{max}, T_{min}, λ, I)

ENTRÉE: v : nombre de segments des itinéraires initiaux
ENTRÉE: w : nombre de segments des itinéraires terminaux
ENTRÉE: t_{min} : le temps minimum d'attente dans un aéroport
ENTRÉE: t_{max} : le temps maximum d'attente dans un aéroport
ENTRÉE: T_{min} : l'information sur les bornes inférieures de temps
ENTRÉE: λ : le rapport maximal de durée d'un itinéraire par rapport au meilleur temps pour une même (O,D)
SORTIE: I est l'ensemble solution mis à jour

1: **POUR TOUT** $a \in \{a \mid (a,b) \in \mathcal{A}_*^2\}$ **FAIRE**
2: $I_L \leftarrow L^+(a,t)$ // la liste des itinéraires de v segments de vol partant de a
3: **POUR TOUT** $i_L \in I_L$ **FAIRE**
4: $a' \leftarrow \mathcal{D}(i_L)$: l'aéroport intermédiaire
5: $I_R \leftarrow L^+(a',u)$ // la liste des itinéraires de w segments de vol décollant de a'
6: **POUR TOUT** $i_R \in I_R$ **FAIRE**
7: $t_1 \leftarrow$ le temps d'atterrissage de i_L, date et heure locale
8: $t_2 \leftarrow$ le temps de décollage de i_R, date et heure locale
9: $w_{a'} \leftarrow t_2 - t_1$, le temps d'attente à l'aéroport a'
10: $b \leftarrow \mathcal{D}(i_R)$: la destination de l'itinéraire terminal
11: **SI** $(t_1 + t_{min} \leq t_2) \wedge (w \leq t_{max}) \wedge (a \neq b)$ **ALORS**
12: $t' \leftarrow T_{min}(a,b) * \lambda$: le plus long temps admissible pour un itinéraire de a vers b
13: **SI** $\mathcal{T}(i_L) + \mathcal{T}(i_R) + w_{a'} \leq t'$ **ALORS**
14: **SI** i_L et i_R ne bouclent pas **ALORS**
15: $i' \leftarrow (i_L + i_R)$: un nouvel itinéraire constitué de i_L puis ensuite de i_R
16: $CONNAITRE(i')$
17: $I \leftarrow I \cup \{i'\}$

Les combinaisons d'itinéraires qui mènent à des itinéraires de plus d'un segment représentant le coeur du problème résolu par l'algorithme 4.3.1 est explicité par l'algorithme 4.3.3 ci-après. En effet, il est plutôt trivial de construire des itinéraires de passagers pour chacun des segments de vol de l'horaire des vols mais il est plus ardu de pouvoir énumérer les itinéraires composés de plusieurs segments de manière efficace.

La ligne 14 de l'algorithme 4.3.3 signifie qu'il faut s'assurer de ne pas construire

un itinéraire qui présente une boucle, c'est-à-dire qu'un aéroport intermédiaire est visité deux fois tel qu'illustré à la figure 4.3.

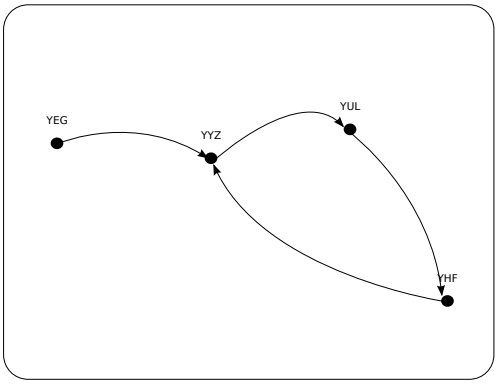

FIGURE 4.3 Exemple d'itinéraire qui boucle

Un nouvel itinéraire est trouvé à la ligne 15 de l'algorithme 4.3.3 en concaténant deux itinéraires existants tel que montré à la figure 4.4.

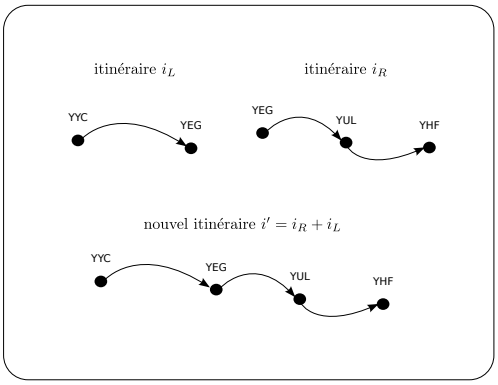

FIGURE 4.4 Création d'un nouvel itinéraire $i' = i_L + i_R$

4.3.2 Complexité temporelle

Nous supposons avoir un ensemble $\mathbb{A}$ d'aéroports où $|\mathbb{A}| = \alpha$ et un ensemble de m segments de vol. Chaque segment de vol donne lieu à un itinéraire de passagers composé d'un segment.

On suppose aussi que tous les itinéraires à un segment sont répartis uniformément sur l'ensemble des aéroports, c'est-à-dire que chaque aéroport comporte le même nombre d'itinéraires à un segment qui y décolle que le nombre d'itinéraires à un segment y atterrissant. On a donc alors que chaque aéroport a m/α itinéraires y décollant et m/α itinéraires y atterrissant. Même chose pour le nombre des itinéraires atterrissant par aéroport.

Au moment de constituer les itinéraires composés de deux segments, on doit considérer

$$\frac{m}{\alpha} \times \frac{m}{\alpha} = \left(\frac{m}{\alpha}\right)^2$$

combinaisons pour un aéroport intermédiaire donné. Comme on doit procéder ainsi pour chacun des aéroports intermédiaires, on aura alors

$$\left(\frac{m}{\alpha}\right)^2 \times \alpha = \frac{m^2}{\alpha}$$

comparaisons à effectuer pour chacun des aéroports intermédiaires donnés. On aura alors, au pire, $m + m^2/\alpha$ itinéraires de un ou de deux segments, s'ils sont tous conservés.

Pour constituer les itinéraires à trois segments, on doit vérifier la connectabilité des itinéraires à deux segments avec les itinéraires à un segment. On doit donc faire

$$\frac{m^2}{\alpha} \times \left(\frac{m}{\alpha}\right)^2 = \frac{m^3}{\alpha^2}$$

comparaisons. Pour chacun des aéroports intermédiaires, nous avons

$$\frac{m^3}{\alpha^2} \times \alpha = \frac{m^3}{\alpha}$$

comparaisons à effectuer. Et on aura alors, au pire, $m + m^2/\alpha + m^3/\alpha$ itinéraires de un, deux ou trois segments. Nous avons alors une complexité temporelle de $O(m^3)$.

Ce scénario nous aura amené à effectuer en moyenne $O((\frac{m}{|\mathbb{A}|})^{k_{max}})$ comparaisons pour des itinéraires d'au plus k_{max} segments de vol pour toutes les paires d'aéroports possibles.

4.4 Optimisation locale

Au moment où des modifications partielles sont apportées à un horaire de vols existant, on souhaite apporter les modifications à l'ensemble des itinéraires générés à partir de cet horaire mais uniquement aux itinéraires affectés. Cela signifie qu'il faut éliminer les itinéraires pour lesquels au moins un des segments de vol a été retiré de l'horaire des vols et aussi ajouter tout nouvel itinéraire qui semble valable en considérant l'un ou l'autre des nouveaux segments de vol.

On doit donc éliminer les itinéraires n'étant plus pertinents et mettre à jour les structures de données concernées. Par la suite, on instantie un nouvel itinéraire pour chacun des nouveaux segments issus de la modification de l'horaire de vols. Finalement, on ne procède qu'en énumérant des itinéraires composés d'au moins un nouvel itinéraire.

Puisqu'on doit constituer uniquement des itinéraires de vol qui comportent au moins un itinéraire nouvellement créé, il sera très utile que chaque nouvel itinéraire soit muni d'une étiquette qui représentera cette caractéristique. Avant de lancer l'énumération des itinéraires pour un horaire de vol donné, il faudra réinitialiser la valeur de ces propriétés à une valeur indiquant que les itinéraires sont issus d'une itération antérieure. L'algorithme 4.4.1 présente la marche à suivre et le marquage des itinéraires issus d'un ancien horaire de vols est indiqué à la ligne 1.

Nous devons donc procéder à la création de nouveaux itinéraires de passagers à partir de chacun des segments de vol nouveaux pour cette itération. L'énumération des itinéraires comportant au moins un nouveau segment est réalisée dans l'algorithme 4.4.2. Notons que, hormis la ligne 1, l'algorithme 4.4.2 diffère de l'algorithme 4.3.3 par la ligne 8. On vérifie alors que le candidat en cours de construction est formé d'au moins un nouvel itinéraire de passagers. Si c'est le cas, on pourra alors conserver ce nouvel itinéraire et automatiquement le marquer comme tel.

En ce qui a trait aux mises à jour de structures de données, notons déjà que les expériences ont révélé qu'il est plus rapide d'effacer chacune des structures de données utilisées dans l'algorithme 4.3.3 et de les reconstituer complètement plutôt que de procéder à des corrections sélectives. En effet, trouver chacune des instances d'itinéraires à éliminer, soit ceux dont au moins un des segments de vol a été retiré de l'horaire des vols, dans chacun des sous-ensembles est plus fastidieux que de construire de nouveaux sous-ensembles à partir de rien.

Algorithme 4.4.1 RECALCUL($\mathbb{S}_m$)

ENTRÉE: $\mathbb{S}_m$: l'ensemble des nouveaux segments de vol à l'itération m
SORTIE: I_m : l'ensemble des nouveaux itinéraires issus des modifications à l'horaire des vols pour l'itération m

1: marquer tous les itinéraires existants comme étant antérieurs à cette itération
2: effacer les structures de données de sous-ensemble par aéroport et par nombre de segments de vol
   $I_m \leftarrow \emptyset$
3: **POUR TOUT** $s \in \mathbb{S}_m$ **FAIRE**
4: i est un itinéraire composé uniquement de s
5: i est marqué comme étant nouveau
6: $I_m \leftarrow I_m \cup \{i\}$
7: CONNAÎTRE(i)
8: **POUR TOUT** $p \in \{2, \ldots, k_{max}\}$ **FAIRE**
9: CONNECTER-NOUVEAUX($p-1$, 1, t_{min}, t_{max}, T_{min}, λ, I_m)
10: **RETOURNER** I_m

Algorithme 4.4.2 CONNECTER-NOUVEAUX(v, w, t_{min}, t_{max}, T_{min}, λ, I_m)

ENTRÉE: v : nombre de segments des itinéraires initiaux
ENTRÉE: w : nombre de segments des itinéraires terminaux
ENTRÉE: t_{min} : le temps minimum d'attente dans un aéroport
ENTRÉE: t_{max} : le temps maximum d'attente dans un aéroport
ENTRÉE: T_{min} : les bornes inférieures de temps entre paires d'aéroports
ENTRÉE: λ : le rapport maximal de durée d'un itinéraire par rapport au meilleur temps entre pour une même (O, D)
SORTIE: I_m : l'ensemble solution mis à jour

POUR TOUT $a \in \{a \mid (a,b) \in \mathcal{A}_*^2\}$ **FAIRE**
  $I_L \leftarrow L^+(a,t)$: la liste des itinéraires de v segments de vol partant de a
  POUR TOUT $i_L \in I_L$ **FAIRE**
    $a' \leftarrow \mathcal{D}(i_L)$: l'aéroport intermédiaire
    $I_R \leftarrow L^+(a',u)$: la liste des itinéraires de w segments de vol décollant de a'
    POUR TOUT $i_R \in I_R$ **FAIRE**
      SI i_L ou i_R ne bouclent pas **ALORS**
        SI i_L ou i_R est un nouvel itinéraire **ALORS**
          $t_1 \leftarrow$: le temps d'atterrissage de i_L, date et heure locale
          $t_2 \leftarrow$: le temps d'atterrissage de i_R, date et heure locale
          $w_{a'} \leftarrow t_2 - t_1$: le temps d'attente à l'aéroport a'
          $b \leftarrow \mathcal{D}(i_R)$: la destination de l'itinéraire terminal
          SI $(t_1 + t_{min} \leq t_2) \wedge (w \leq t_{max}) \wedge (a \neq b)$ **ALORS**
            $t' \leftarrow T_{min}(a,b) * \lambda$: le plus long temps admissible pour de a vers b
            SI $\mathcal{T}(i_L) + \mathcal{T}(i_R) + w_{a'} \leq t'$ **ALORS**
              SI i_L et i_R ne bouclent pas **ALORS**
                Un nouvel itinéraire est créé
                $i' \leftarrow (i_L + i_R)$: un nouvel itinéraire constitué de i_L puis ensuite de i_R
                $I_m \leftarrow I_m \cup \{i'\}$
                $CONNAITRE(i')$

Chapitre 5

RÉSULTATS

Ce chapitre présente quelques résultats numériques qui ont été obtenus à l'aide d'implémentations des algorithmes présentés au chapitre 4. Nous y verrons une description sommaire des données utilisées pour composer un modèle d'essai dans la section 5.2. Ensuite, la section 5.3 présente les critères monitorés qui visent à évaluer la performance relative des algorithmes implémentés. La section présentera finalement des résultats numériques en commençant par une comparaison des différentes méthodes puis une analyse de sensibilité pour la méthode trouvée la plus performante et finalement des résultats en rapport avec les re-calculs lors de modifications successives à un horaire de vols.

Le lecteur trouvera tout d'abord une description des conditions de simulations numériques dans lesquelles les résultats numériques ont été obtenus et présentés dans ce chapitre à la section 5.1.

On peut trouver dans la section B des tableaux de données pour évaluer la sensibilité des algorithmes qui ne sont pas traités dans le présent chapitre.

5.1 Banc de simulations

Les algorithmes présentés dans ce mémoire ont été implémentés en C++ et compilés avec GNU G++ v4.3.2.

Les calculs ont été réalisés sur une machine dotée d'un processeur 64 bits à quatre coeurs d'Intel ©. Le processeur est cadencé à 2,83 GHz avec 4 Go de mémoire vive. Le système d'exploitation utilisé était OpenSUSE 11.1 (x86-64).

5.2 Jeu de données

Les données utilisées pour décrire le réseau de transport aérien sont des données de prévisions internes de la compagnie Air Canada pour sa saison d'été 2005. Plus pré-

cisément, les données disponibles sont celles produites par un logiciel de planification opérationnelle utilisé chez Air Canada. Ce logiciel est produit par Lufthansa Systems et fait partie de la suite informatique Netline/Ops (voir `http://www.lhsystems.com/`). Cette suite logicielle propose des outils pour la gestion des opérations de lignes aériennes. Les données historiques réelles représentent des données sensibles pour la compagnie aérienne et n'ont pu être obtenues. Les données de Netline pour l'été 2005 représentent une semaine typique du réseau de transport aérien d'Air Canada en termes de vols de passagers. En effet, ces données représentent des itinéraires de passagers. Chaque itinéraire de passagers est décrit par :

1. aéroport de départ (ou origine) ;
2. aéroport de destination ;
3. heure locale de départ à l'origine ;
4. heure locale d'arrivée à la destination ;
5. journée de départ, pouvant être du lundi au dimanche inclusivement ;
6. durée totale de l'itinéraire en minutes ;
7. type d'appareil assurant le trajet (Boeing 343, Airbus A320, etc.) ;
8. demande sans contraintes ;
9. demande avec contraintes ;
10. nombre de segments de vol, ou vols, constituant l'itinéraire ;
11. numéro de vol pour chacun des vols qui composent chaque itinéraire ;
12. heure de départ de chacun des vols intermédiaires s'il y a lieu ;
13. jour de départ de chacun des vols intermédiaires s'il y a lieu ;
14. jour d'arrivée de l'itinéraire.

La demande sans contraintes représente la demande des passagers alors que la demande sous contraintes représente la demande qui a été limitée par la capacité des appareils. Ces données d'entrée représentent des itinéraires qu'il sera possible de décomposer en segments de vol sur un horaire cyclique d'une semaine. En tout, ce sont 117 551 itinéraires de passagers qui sont décrits dans le jeu de données. De ces itinéraires, on peut extraire 10 490 segments de vol, ou vols, distincts.

5.3 Critères de performance

Pour chaque paire (O, D), on peut déterminer une demande totale qui est la somme des passagers ayant voyagé de O vers D au courant d'une semaine selon l'information disponible dans les données de Netline . C'est donc la somme du nombre de passagers ayant voyagé sur tout itinéraire de O vers D durant la semaine, soit

$$Demande(O, D) = \sum_{i \in I(O,D)} nbPassagers(i)$$

où $I(O, D) = \{$itinéraires reliant O à $D\}$ où $nbPassagers(i)$ donne le nombre de passagers ayant emprunté l'itinéraire i tel que mentionné dans les données de Netline. Pour chaque itinéraire de l'ensemble d'itinéraires de passagers énuméré par l'un ou l'autre des algorithmes et qui se retrouve dans les données de Netline, on pourra considérer que le nombre de passagers correspondant dans Netline aura été desservi par le nouvel ensemble. Ainsi, pour chaque paire (O, D), on pourra estimer une couverture de service et quantifier la proportion de marché desservie par le nouvel ensemble d'itinéraires.

Le tableau 5.1 présente les caractéristiques de performance qu'on tentera d'évaluer.

Critère	Unités	Description
Temps de calcul	**secondes**	Temps requis à l'algorithme pour générer les itinéraires de passagers pour toutes les paires (O, D) où il y a de la demande (au moins un passager qui souhaite voyager de O vers D).
Nombre d'itinéraires générés	**itinéraires**	Cardinalité de l'ensemble d'itinéraires énumérés.
Proportion de couverture pour l'échantillon	%	La moyenne des pourcentages de couverture pour l'échantillon présenté au tableau 5.2.
Proportion de couverture pour toutes les paires (O, D)	%	En se référant au jeu de données de Netline, on considère que tous les itinéraires générés qui se retrouvent dans les données de Netline auraient pu desservir le même nombre de personnes qu'indiqué pour l'itinéraire de Netline.
Couverture $< 85\%$	%	C'est la moyenne de toutes les proportions de couverture de service inférieures ou égales à 85%, toutes paires (O, D) confondues.
Proportion de marché «mal desservie»	%	La proportion de la demande des toutes les paires (O, D) mal servies par rapport à la demande totale de toutes les (O, D). On estime qu'une paire (O, D) est mal desservie si la couverture de services est inférieure ou égale à 85% de la demande. La proportion est exprimée en pourcentage.

TABLEAU 5.1 Critères de performance

On considère qu'un paire origine et destination (O, D) est mal desservie si elle est en deçà de 85% de couverture relative par rapport aux données de Netline pour l'été 2005.

On peut déjà songer que l'énumération exhaustive de tous les itinéraires possibles, peu importe les critères d'acceptabilité utilisés, pourrait résoudre le problème de la couverture de service et réduire à zéro les marchés mal desservis. Mais ceci aurait pour effet de congestionner le modèle de flots de passagers qui utiliserait ces itinéraires. Il faudrait donc choisir des valeurs de paramètres qui soient judicieux. Nous espérons pouvoir trouver des indications dans les indicateurs de performance qui pourraient nous aider à porter un jugement sur les ensembles d'itinéraires énumérés par les algorithmes.

5.3.1 Échantillon témoin

Il y a un échantillon témoin de paires (O, D) pour lesquelles certains critères particuliers de performance (ou de qualité) seront comptabilisés. Cet échantillon comporte les douze (12) paires (O, D) présentées dans le tableau 5.2. Les codes à trois lettres sont des identifiants d'aéroports publiés par l'Association Internationale du Transport Aérien (AITA). La correspondance entre chaque code à trois lettres et l'aéroport est décrite dans le tableau A.1 de l'annexe.

Portée géographique	**Paires** (O, D)
Vols domestiques	(YBC, YUL), (YYZ, YLW), (YWK, YOW), (YYC, YHZ)
Vols intra-continentaux	(YYZ, BGI), (PDX, YHZ), (YQM, BWI), (BOS, YUL)
Vols extra-continentaux	(YUL, CDG), (YOW, LHR), (YEG, PEK), (HNL, YVR)

TABLEAU 5.2 Échantillon de paires (O, D)

Les données historiques disponibles sont les itinéraires pour la saison été de 2005 chez Air Canada. Il est donc naturel de voir à déterminer un échantillon représentatif des marchés que peut viser ce transporteur aérien. Cet échantillon a été constitué de manière à pouvoir représenter des itinéraires générés pour les trois types de marchés suivants :

1. domestiques, c'est-à-dire des vols ayant cours à l'intérieur du Canada seulement ;
2. internationaux intra-continentaux, c'est-à-dire des vols internationaux ayant cours aux Amériques seulement ;
3. internationaux extra-continentaux, c'est-à-dire des vols internationaux ayant cours entre l'Amérique et l'Europe ou l'Asie.

5.4 Résultats numériques

Voici donc les résultats obtenus pour chacune des méthodes d'énumération des itinéraires. Tout d'abord, la section 5.4.1 présente une comparaison des critères de performance généraux pour chacun des algorithmes décrits dans le chapitre 4, chacun dans sa forme avec pré-traitement et avec mise-à-jour dynamique des bornes inférieures de temps de voyage. Ensuite, nous présenterons dans la section 5.4.2 une

analyse de sensibilité au sujet de la méthode qui aura montré les meilleurs indicateurs de performance. Finalement, la section 5.4.3 discute des temps d'exécution et du nombre d'itinéraires générés lors de ré-optimisations locales.

5.4.1 Comparaisons des différentes méthodes

Pour énumérer les itinéraires qui sont susceptibles d'intéresser une clientèle de passagers, on utilise les critères énoncés dans la section 3.1.2. Dans le cadre qui nous concerne particulièrement, chacune de ces contraintes sera paramétrisable au moment d'exécuter l'algorithme visant à énumérer les itinéraires de passagers. Ainsi, il sera possible de déterminer les impacts sur les temps de calcul et d'évaluer la qualité de l'ensemble solution obtenu pour chaque combinaison de valeurs des paramètres. Rappelons les paramètres décrits dans la section 3.1.2 :

1. un itinéraire de passagers est constitué d'un nombre maximum de segments de vol et nous identifierons ce paramètre par k_{max} ;
2. le temps de l'itinéraire ne doit pas dépasser $\lambda \times T_{min}(O, D)$, où $\lambda \in \mathbb{R}_*^+$ est un nombre réel positif ;
3. dans un itinéraire composé de plus d'un segment, un temps minimum est requis entre l'arrivée d'un segment et le départ du segment suivant, qui sera identifié par t_{min} ;
4. dans un itinéraire composé de plus d'un segment, un temps maximum d'attente pour un passager à un aéroport avant d'entreprendre des déplacements sur un autre segment, identifié par t_{max}.

Chacun de ces paramètres aura une valeur par défaut tel que présenté dans le tableau 5.3. Ce sont ces valeurs qu'il faudra considérer lors des simulations numériques sauf avis contraire.

Paramètre	**Valeur**
k_{max}	3
t_{min}	30
t_{max}	180
λ	2.0

TABLEAU 5.3 Valeurs par défaut des paramètres d'exécution

Notons que les méthodes dynamiques visent à éliminer un pré-traitement qui permet de recueillir des valeurs de bornes inférieures que nous obtenons par l'exécution d'un algorithme des plus courts chemins. Dans notre cas particulier, nous avons utilisé une implémentation de l'algorithme de Dijkstra. Pour le réseau construit à partir de l'horaire de vols décrits dans les données de Netline, ce travail a représenté un temps d'environ 35 secondes sur le système informatique utilisé. Dans chacun des temps montrés ci-après, aucun n'inclus le temps de pré-traitement. Il faut donc penser que les méthodes de recherche en profondeur et combinatoire auront dû prendre ce temps de plus pour le pré-traitement.

Le tableau 5.4 présente les résultats obtenus par les quatre méthodes proposées pour le problème complet.

Critères de performance	**Methodes d'énumération**			
	Recherche en profondeur	Recherche en profondeur dynamique	Combinatoire	Combinatoire dynamique
Temps (sec)	2910.4	1400.6	13.2	17.5
Nombre d'itinéraires	399618	141884	344625	498110
Couverture échantillon (%)	87.8	59.8	87.8	87.8
Couverture globale (%)	94.8	79.9	94.7	96.1
Couverture $<$ 85% (%)	61.9	60.7	62.2	52.7
Proportion de marché «mal desservi» $<$ 85% (%)	8.5	13.1	8.6	4.9

TABLEAU 5.4 Comparaisons de performances des méthodes

Au premier coup d'oeil, on remarque tout de suite que les temps d'exécution pour les méthodes de recherche en profondeur sont environ deux ordres de grandeur plus grands que pour les méthodes combinatoire, même si on ajoute environ 35 secondes de pré-traitement pour la méthode combinatoire non-dynamique. C'est principalement dû au fait que des sous-chemins dans le réseau sont parcourus à plusieurs reprises par les méthodes de recherche de recherche en profondeur, dans différentes amorces de chemins, comme montré dans la figure 5.1.

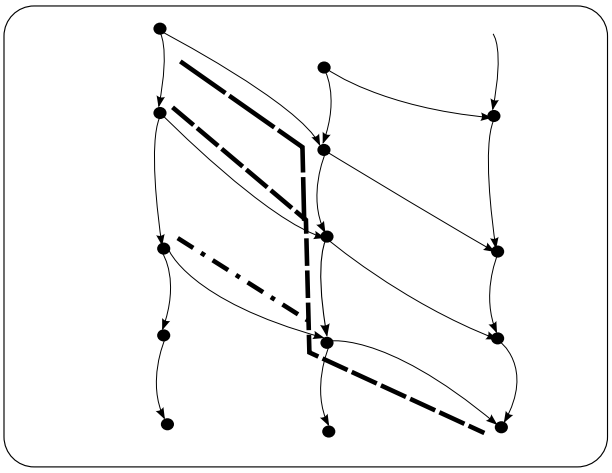

FIGURE 5.1 Parcours redondants

On explique aussi ces différences de temps par le fait que l'algorithme doit composer avec la présence d'arcs d'attente qui sont des arcs ajoutés entre deux évènements de décollage ou d'atterrissage dans un même aéroport.

On remarque aussi que les deux méthodes sans mise-à-jour dynamique des bornes inférieures de temps ne donnent pas le même nombre d'itinéraires de vols. Une comparaison des ensembles d'itinéraires énumérés par ces méthodes aura permis de constater que l'ensemble des itinéraires trouvés par la méthode combinatoire (C), disons $\mathcal{C}$, est inclus dans l'ensemble des itinéraires trouvées par la méthode de recherche en profondeur (RP), disons $\mathcal{RP}$. Formellement, on a que $\mathcal{C} \subset \mathcal{RP}$

En observant les itinéraires se trouvant dans la différence $\mathcal{RP} \setminus \mathcal{C}$, on trouve un itinéraire comme le suivant :

Vol	**Origine**	**Destination**	**Jour**	**Décollage**	**Atterrissage**	**Durée (min.)**
AC 1238	YQG	YYZ	jeudi	06h13	07h15	62
AC 446	YYZ	YOW	jeudi	10h00	11h01	61
AC 3973	YOW	YWG	jeudi	11h45	13h39	174

TABLEAU 5.5 Exemple d'itinéraire trouvé avec la méthode RP mais pas avec C

La durée totale de cet itinéraire constitué de 3 segments de vol est de 506 minutes alors que le temps maximal pour un itinéraire reliant l'aéroport YQG à l'aéroport YWG est de 512 minutes au plus. Les temps d'attente sont respectivement de 174

et 44 minutes, ce qui est bien dans l'intervalle fermé $[t_{min} = 30,\ t_{max} = 180]$. Mais comment se fait-il que cet itinéraire soit pas trouvé par la méthode combinatoire ? En regardant dans le jeu de données, on voit que le préfixe de cet itinéraire, constitué des vols AC 1238 et AC 446 n'est pas trouvé par la méthode combinatoire non plus. On constate que le temps maximal admissible pour voyager de YYZ vers YWG est de 272 minutes mais que les vols AC 1238 et AC 446 s'exécutent en 288 minutes lorsqu'on inclut le temps d'attente à l'aéroport YYZ, ce qui explique la raison de son rejet. L'itinéraire décrit dans le tableau 5.5 montre donc un cas où un itinéraire qui soit globalement admissible mais qui est constitué de sous-itinéraires qui sont inadmissibles localement.

Résumons comment les deux méthodes se comportent au moment de chercher un itinéraire de passagers qui soit composé de plusieurs segments de vol. Supposons que nous ayons un itinéraire I' composé de trois vols qui visite successivement les aéroports $A,\ B,\ C,\ D$. La méthode combinatoire construit des itinéraires dont tous les préfixes respectent les critères d'admissibilité dans le contexte où ils s'appliquent. C'est une conséquence de la définition récursive 4.3.1 qu'elle utilise. Nous savons déjà que le préfixe de I' partant de A vers B sera admissible car c'est un vol direct. Le préfixe de I' allant de A vers C produira un nouvel itinéraire admissible si, en combinant deux itinéraires jugés admissibles individuellement, est aussi admissible. Mais la méthode de recherche en profondeur cherche un chemin, entre A et D, peu importe les sous-chemins qui sont impliqués. On a donc que RP effectue plus de comparaisons que C pour une même paire (O, D).

La variante dynamique de la méthode combinatoire énumère plus d'itinéraires que la méthode combinatoire avec pré-traitement car les critères d'acceptation d'un itinéraire sont plus laxistes au début et se raffinent au courant de l'exécution de l'algorithme alors que les valeurs numériques sont relaxées. Par contre, la méthode de recherche en profondeur dynamique trouve moins d'itinéraires que la méthode de recherche en profondeur avec pré-traitement. C'est dû au fait que certaines valeurs de la fonction h de l'équation (4.1) sont parfois arbitrairement grandes lorsqu'aucune relaxation n'a pu être effectuée, ce qui a pour effet de cesser la recherche dans certaines directions.

Ensuite, on peut observer que les deux méthodes combinatoires présentent une plus grande uniformité au niveau des temps d'exécution et du nombre d'itinéraires énumérés que pour les méthodes de recherche en profondeur.

Les méthodes de recherche en profondeur sont beaucoup plus sensibles à la disponibilité d'un pré-traitement qui soit mieux renseigné que des bornes inférieures qui soient relaxées en cours de route. On peut expliquer cela par le fait que les méthodes combinatoires commencent par énumérer tous les itinéraires à un vol, ce qui fournit déjà des valeurs de bornes inférieures pour plusieurs (O, D), pour ensuite énumérer les itinéraires à deux vols et fixer certaines valeurs de bornes inférieures pour des (O, D) dont les meilleurs itinéraires disponibles comptent deux segments de vol, et ainsi de suite. Mais les méthodes de recherche en profondeur peuvent trouver des itinéraires à plusieurs segments de vols au fur et à mesure que des itinéraires à un seul segment de vol sont trouvés, ce qui augmente le nombres de chemins amorcés mais qui sont moins bons ; l'émondage permis par l'heuristique est moins efficace dans de telles conditions.

De plus, comme la méthode combinatoire procède par l'agencement d'itinéraires existants, le temps total de l'itinéraire en cours de confection est connu immédiatement par la somme des temps de chacun des itinéraires et du temps d'attente au moment de passer d'un itinéraire à l'autre. Dans le cas des algorithmes de recherche en profondeur, c'est une heuristique qui contribue à déterminer une borne inférieure sur le temps restant. La méthode combinatoire ne souffre pas de la présence d'arcs d'attente ; la structure de données qui représente un réseau comporte au moins un arc d'attente pour chaque arc implicite.

La méthode combinatoire dynamique énumère beaucoup d'itinéraires, ce qui peut expliquer la raison de son rendement en termes de couverture de service : le meilleur taux de couverture globale ainsi que la plus faible proportion de marchés mal desservis. Mais un ensemble d'itinéraires qui soit de grande cardinalité posera problème au modèle de flots de passagers. Le court temps d'exécution permet de présager qu'il serait raisonnable d'implémenter un post-traitement sur l'ensemble d'itinéraires afin d'éliminer certains itinéraires pouvant être inutiles si on en compte un grand nombre pour une paire (O, D) donnée, par exemple.

5.4.2 Analyse de sensibilité

Les paramètres décrits dans le tableau 5.3 déterminent l'acceptation ou le rejet d'itinéraires de passagers durant le processus d'énumération. Ces paramètres présentent donc une forme de contrôle sur la qualité de l'ensemble d'itinéraires trouvés. Ainsi, il semble naturel de considérer la sensibilité relative de chacun de ces paramètres

pour différentes valeurs.

Nous ferons varier la valeur de chacun de ces paramètres à la fois en laissant les autres à leurs valeurs par défaut. Pour chacun des paramètres particuliers, nous aurons ces gammes de valeurs :

- $\lambda \in \{1.5,\ 2.0,\ 2.5,\ 3.0,\ 3.5\}$;
- $k_{max} \in \{1,\ 2,\ 3,\ 4\}$;
- $t_{min} \in \{30,\ 60,\ 90\}$;
- $t_{max} \in \{120,\ 150,\ 180,\ 210,\ 240\}$.

Fort des résultats présentés dans la section précédente, nous pensons que la méthode combinatoire dynamique présente le meilleur choix parmi les quatre méthodes car elle offre des performances similaires à la méthode combinatoire mais élimine les 35 secondes requises pour le pré-traitement tout en n'augmentant que d'une seconde le temps de calcul. Les sections qui suivent présentent les résultats obtenus par variation de chacun des paramètres d'exécution pour la méthode combinatoire dynamique.

Facteur multiplicatif de la borne inférieure

Les tableaux 5.6 et 5.7 présentent les critères de performance selon la variation du paramètre λ. Notons que la colonne du tableau 5.6 avec le facteur 2.0 est montrée dans le tableau 5.4 présenté précédemment.

Critères de performance	λ				
	1.5	2.0	2.5	3.0	3.5
Temps (sec)	14.6	17.5	18.3	18.9	19.3
Nombre d'itinéraires	289148	498110	593575	644009	672153
Couverture échantillon (%)	87.8	87.8	87.8	87.8	87.8
Couverture globale (%)	95.4	96.1	96.1	96.1	96.1
Couverture $<$ 85% (%)	57.0	52.7	52.6	52.6	52.6
Proportion de marché «mal servi» $<$ 85% (%)	6.4	4.9	4.9	4.9	4.9

TABLEAU 5.6 Performances pour différentes valeurs de λ, méthode Combinatoire dynamique

Paire (O, D)	λ					
	1.5	2.0	2.5	3.0	3.5	Netline
(YBC, YUL)	7	7	7	7	7	7
(YYZ, YLW)	280	774	844	844	844	142
(YWK, YOW)	6	6	6	6	6	6
(YYC, YHZ)	475	1337	1379	1386	1393	172
(YYZ, BGI)	7	7	7	7	7	7
(PDX, YHZ)	131	131	131	131	131	7
(YQM, BWI)	38	38	38	38	38	21
(BOS, YUL)	165	304	562	731	802	51
(YUL, CDG)	111	219	307	318	318	46
(YOW, LHR)	291	604	1052	1073	1122	83
(YEG, PEK)	66	72	72	80	101	7
(HNL, YVR)	11	11	11	11	11	11

TABLEAU 5.7 Nombre d'itinéraires générés, variations du paramètre λ, méthode Combinatoire dynamique

Considérons tout d'abord le tableau 5.6. Les valeurs de pourcentages, autant en ce qui concerne la couverture de service de l'échantillon que la couverture de service globale, ne sont que peu ou pas influencées par rapport à l'augmentation du nombre d'itinéraires trouvés car des itinéraires nouvellement énumérés ne se retrouvent pas dans les données de Netline qui comportent 117 551 itinéraires disponibles. Mais nous en avons énuméré de 289 148 à 672 153 selon les diverses valeurs du paramètre λ. Comme le pourcentage de recouvrement représente la proportion de passagers qui sont desservis parmi les itinéraires de Netline, nous ne pouvons estimer le nombre de passagers qui pourraient être enclins à utiliser un nouvel itinéraire ne figurant pas dans les données historiques. On ne peut donc pas quantifier les itinéraires ne se trouvant pas dans les données de Netline.

Mais il semble que la valeur du facteur λ doit être égale ou plus grande à 2 car on passe de 95.4% à 96.1% de couverture globale à partir de cette valeur de paramètre. La couverture globale plafonne à partir de cette valeur. On peut observer un comportement similaire pour les autres métriques montrées.

Le tableau 5.7 nous présente quelques cas qui ne sont pas influencés par le facteur de borne inférieure, soient (YBC, YUL), (YWK, YOW), (YYZ, BGI), (YQM, BWI) et (HNL, YVR). Après vérifications, toutes ces paires d'aéroports sont desservies

par des vols directs. Par exemple, le dernier cas entre Honolulu (HNL) et Vancouver (YVR) présente un ensemble d'arcs qui constitue un séparateur du réseau espace-temps. Il en va de même entre Baie Comeau (YBC) et Montréal (YUL) et aussi pour Wabush (YWK) et Ottawa (YOW).

Pour une paire d'origine et destination comme Calgary (YYC) et Halifax (YHZ) par exemple, on observe une grande fluctuation du nombre d'itinéraires trouvés selon le facteur utilisé car ces aéroports peuvent être rejoints par bon nombre d'aéroports intermédiaires, comme Montréal (YUL) et Toronto (YYZ) par exemple, ce qui permet un grand nombre de possibilités de connexions pour des passagers.

Autre fait à remarquer, une paire d'aéroports qui compte 7 itinéraires signifie habituellement qu'un même itinéraire est disponible pour chacun des jours de la semaine ; rappelons que l'horaire des vols est cyclique sur un horizon de temps d'une semaine.

Pour la paire d'origine et destination Portland (PDX) et Halifax (YHZ), on trouve dans les données de Netline seulement 7 itinéraires pouvant les relier alors que notre méthode en trouve un nombre aussi grand que 131. Après inspection des itinéraires de Netline, cette paire d'aéroports ne se voit être desservie que par des itinéraires à deux segments, à raison d'un itinéraire par jour, d'où 7 itinéraires en tout. Mais la méthode combinatoire dynamique en trouve bon nombre qui sont composés de trois segments de vols et dont les critères d'acceptation sont convenables par rapport à nos critères donnés. On peut facilement penser que la proximité relative de ces deux aéroports incite les passagers à ne choisir que les itinéraires à 2 segments de vols.

Nombre de segments de vol

Les tableaux 5.8 et 5.9 présentent les résultats obtenus en fonction des valeurs que peut prendre le paramètre k_{max}. On y observe clairement que les données de Netline comportent 10 490 vols, ou segments de vol, distincts.

Critères de performance	k_{max}			
	1	2	3	4
Temps (sec)	0.0	2.9	17.5	53.0
Nombre d'itinéraires	10490	138012	498110	1034447
Couverture échantillon (%)	49.6	73.4	87.8	87.8
Couverture globale (%)	69.5	95.2	96.1	96.1
Couverture $<$ 85% (%)	63.2	57.2	52.7	52.1
Proportion de marché «mal servi» $<$ 85% (%)	15.6	5.5	4.9	5.0

TABLEAU 5.8 Performances pour différentes valeurs de k_{max}, méthode Combinatoire dynamique

Paire (O, D)	k_{max}				
	1	2	3	4	Netline
(YBC, YUL)	7	7	7	7	7
(YYZ, YLW)	1	192	774	1154	142
(YWK, YOW)	0	0	6	105	6
(YYC, YHZ)	7	190	1337	2779	172
(YYZ, BGI)	7	7	7	7	7
(PDX, YHZ)	0	0	131	512	7
(YQM, BWI)	0	7	38	71	21
(BOS, YUL)	39	218	304	304	51
(YUL, CDG)	14	57	219	219	46
(YOW, LHR)	7	123	604	663	83
(YEG, PEK)	0	14	72	106	7
(HNL, YVR)	11	11	11	11	11

TABLEAU 5.9 Nombre d'itinéraires générés, variations du paramètre k_{max}, méthode Combinatoire dynamique

Le tableau 5.8 nous montre l'impact du nombre maximum de segments de vols que peuvent contenir les itinéraires de passagers énumérés. On voit déjà que les itinéraires à un vol supportent à eux seuls environ presque 70% de la clientèle d'Air Canada durant l'été 2005. On peut trouver un nombre prohibitif d'itinéraires de passagers lorsqu'on admet les itinéraires à 4 segments de vols ; on peut alors compter 1 034 447 itinéraires à 1, 2, 3 ou 4 segments en tout. Mais les données de Netline ne présentent aucun itinéraire à 4 segments ; on n'y retrouve que des itinéraires à au plus 3 segments.

On ne peut donc pas quantifier de tels itinéraires.

De plus, la couverture globale semble plafonner à 96.1%. Et c'est à cause d'itinéraires particuliers, comme ceux pour répondre à des besoins de transports particuliers du genre de Baie Comeau (YBC) et Sept-Îles (YVZ) ou Wabush (YWK) et Ottawa (YOW) par exemple. Ces cas ne sont pas représentés dans les itinéraires énumérés à cause des caractéristiques bien particulières qu'elles représentent ; ce sont des origines et destinations qui sont desservies à la manière d'une route d'autobus. Par là, on entend que le transport est assuré par un petit appareil (DH8) et qu'une série d'aéroports est parcourue par le même appareil, ne faisant que des arrêts très brefs à chacun des aéroports. C'est aussi ce que présente le seuil minimal de la moyenne des itinéraires mal couverts, soient ceux qui sont desservis à moins de 85% ; on ne peut faire mieux qu'une moyenne de 52.1% pour ces (O, D) négligées malgré le nombre très élevé d'itinéraires énumérés. Là aussi, le grand nombre d'itinéraires générés non quantifiés peut expliquer qu'on ne semble pas répondre adéquatement aux besoins de la clientèle en général par d'autres itinéraires.

Le tableau 5.9 montre aussi que la couverture de l'échantillon connaît un nombre excessif d'itinéraires pour transporter les passagers. La variation du paramètre k_{max} ne semble pas pouvoir faire augmenter la métrique de couverture de l'échantillon du tableau précédent.

Le tableau 5.9 corrobore l'intuition que les arcs reliant les paires d'aéroports (YBC, YUL) et (YYZ, BGI) et (HNL, YVR) forment chacun un séparateur du graphe.

Temps minimum de connexion

Les différentes valeurs que peut prendre le paramètre t_{min} influent sur les critères de performance des itinéraires de passagers tels que montré dans les tableaux 5.10 et 5.11.

Notons que la colonne du tableau 5.10 avec le temps minimum de connexion à 30 minutes est montrée dans le tableau 5.4 présenté précédemment.

Critères de performance	t_{min}		
	30	60	90
Temps (sec)	17.5	14.0	10.4
Nombre d'itinéraires	498110	348783	216838
Couverture échantillon (%)	87.8	70.4	65.9
Couverture globale (%)	96.1	91.3	83.6
Couverture $<$ 85% (%)	52.7	58.5	51.6
Proportion de marché «mal servi» $<$ 85% (%)	4.9	15.5	28.9

TABLEAU 5.10 Performances pour différentes valeurs de t_{min}, méthode Combinatoire dynamique

Paire (O, D)	t_{min}			
	30	60	90	Netline
(YBC, YUL)	7	7	7	7
(YYZ, YLW)	774	468	303	142
(YWK, YOW)	6	0	0	6
(YYC, YHZ)	1337	870	503	172
(YYZ, BGI)	7	7	7	7
(PDX, YHZ)	131	68	42	7
(YQM, BWI)	38	34	13	21
(BOS, YUL)	304	207	254	51
(YUL, CDG)	219	193	129	46
(YOW, LHR)	604	439	257	83
(YEG, PEK)	72	46	20	7
(HNL, YVR)	11	11	11	11

TABLEAU 5.11 Nombre d'itinéraires générés, variations du paramètre t_{min}, méthode Combinatoire dynamique

Le tableau 5.10 montre que l'augmentation du temps minimum requis pour effectuer une connexion réduit le nombre de connexions possibles et, conséquemment, le temps de calcul.

Nous observons dans le tableau 5.11 que Wabush et Ottawa (YWK, YOW) ne peut être desservie que par des itinéraires dont le temps de connexion est de 30 minutes ou moins ; au-delà de cette valeur, il est impossible de générer un itinéraire pour cette paire origine et destination. Ça s'explique par le type d'itinéraire qu'on y trouve.

Dans le jeu de données de Netline, on trouve un itinéraire qui relie ces aéroports. Les aéroports visités durant l'itinéraire sont Sept-Îles (YZV) et Québec (YQB). L'appareil utilisé pour effectuer ce trajet est un DH1 (DH8-100) de Bombardier qui peut prendre à son bord un peu plus de 30 passagers.

Pour les itinéraires reliant Moncton (YQM) et Baltimore (BMI), il semble que la majeure partie des connexions disponibles soient pour des valeurs de $0 < t_{min} \leq 60$. Après observations, les itinéraires dans Netline sont composés de deux vols et les temps de connexion sont de 62 minutes, de 182 minutes ou encore de 274 minutes. Les itinéraires avec un grand temps d'attente surviennent à l'heure d'un repas ; vers l'heure du midi dans le cas où y a 182 minutes d'attente ; vers l'heure du souper pour le cas alors qu'il y a 274 minutes d'attente. Ces situations semblent répondre à des impératifs humains : se nourrir. Autre observation : les appareils utilisés affectés à ces itinéraires sont de type *regional jet* et il y a fort à penser que les services de repas ne sont pas disponibles dans ces petits appareils comme c'est le cas avec de plus gros appareils tels que les Boeing 747 ou Airbus 330 par exemple.

L'argument à l'effet que des vols survenant vers l'heure d'un repas offrent un grand temps de connexion pour laisser le passager prendre un repas ne semble cependant pas faire loi car les itinéraires entre Calgary (YYC) et Halifax (YHZ) du jeu de données de référence sont tous constitués de deux segments de vols. Malgré que des connexions aient lieu entre 13h59 et 16h05 à Toronto, cela ne semble pas concorder avec l'heure d'un repas. De plus, les appareils utilisés pour ces itinéraires sont de gros Airbus 320 ; ces appareils sont vraisemblablement munis d'équipements permettant le service de repas aux passagers qu'ils transportent. Il semble alors qu'une autre contrainte qui ne transparaît pas dans le jeu de données exige une telle situation. Peut-être est-ce simplement une raison de trafic aérien.

Les vols reliant Toronto (YYZ) et Kelowna (YLW) ne requièrent un temps de connexion qui soit pas plus long qu'une heure. Il y a même des vols directs entre ces aéroports.

Pour les trajets internationaux, survenant dans de gros aéroports internationaux comme à Londres (LHR) et Ottawa (YOW), un temps de connexion de l'ordre de 2 heures (120 minutes) semble être la norme. Il faut laisser l'opportunité aux passagers de se déplacer dans ces vastes aéroports. De plus, des marges d'erreur sont bienvenues étant donné la longue distance que parcourent les appareils s'y rendant. Plus un vol parcourt une grande distance, et qu'il dure longtemps, plus il y a place à des

retards qui peuvent mettre les transporteurs fautifs dans l'embarras et les obliger à dédommager les passagers ayant manqué leur vol.

De manière générale, on peut dire que l'utilisation de ce seul paramètre ne semble pas permettre de générer un nombre d'itinéraires qui soit convenable car dans la majorité des cas sauf pour Wabush et Ottawa ainsi que pour Baltimore (BWI) et Moncton (YQM), le nombre d'itinéraires reste soit inchangé pour les itinéraires à vols directs, soit excessivement grands pour les autres, si on considère que les données de Netline présentent une indication du nombre d'itinéraires à atteindre. Cette section nous laisse donc entrevoir que les temps minimum de connexion devraient être ajustés à chaque type de vol, voire même à chaque aéroport.

Attente maximale pour connexion

On obtient des valeurs de performance telles que montrées dans les tableaux 5.12 et 5.13 pour les diverses valeurs que peut prendre le paramètre t_{max}.

On remarque ici aussi que la colonne du tableau 5.12 avec le temps d'attente maximum à 180 minutes est montrée dans le tableau 5.4.

Critères de performance	t_{max}				
	120	150	180	210	240
Temps (sec)	11.6	14.7	17.5	20.1	22.6
Nombre d'itinéraires	274422	383406	498110	596314	686196
Couverture échantillon (%)	78.4	87.5	87.8	90.2	92.4
Couverture globale (%)	89.8	94.0	96.1	96.7	97.1
Couverture $<$ 85% (%)	57.2	55.4	52.7	51.4	50.5
Proportion de marché «mal servi» $<$ 85% (%)	16.9	8.7	4.9	3.8	3.2

TABLEAU 5.12 Performances pour différentes valeurs de t_{max}, méthode Combinatoire dynamique

Paire (O, D)	t_{max}					
	120	150	180	210	240	Netline
(YBC, YUL)	7	7	7	7	7	7
(YYZ, YLW)	445	576	774	891	981	142
(YWK, YOW)	6	6	6	6	6	6
(YYC, YHZ)	618	973	1337	1734	1942	172
(YYZ, BGI)	7	7	7	7	7	7
(PDX, YHZ)	49	89	131	179	221	7
(YQM, BWI)	25	31	38	57	73	21
(BOS, YUL)	245	270	304	329	350	51
(YUL, CDG)	131	174	219	252	316	46
(YOW, LHR)	306	438	604	776	952	83
(YEG, PEK)	33	53	72	90	117	7
(HNL, YVR)	11	11	11	11	11	11

TABLEAU 5.13 Nombre d'itinéraires générés, variations du paramètre t_{max}, méthode Combinatoire dynamique

Ici aussi, le tableau 5.13 corrobore l'intuition que les arcs reliant les paires d'aéroports (YBC, YUL) et (YYZ, BGI) et (HNL, YVR) forment chacun un séparateur du graphe. On peut en dire autant de (YWK, YOW).

On peut voir dans le tableau 5.12 qu'un temps d'attente maximum fait croître le nombre d'itinéraires énumérés et ça semble très naturel : la fenêtre temporelle à l'intérieur de laquelle les connexions sont admissibles étant plus grande, on permet un plus grand nombre de connexions. Bien entendu, le temps total maximum que peut prendre un itinéraire ne doit pas dépasser la limite permise et ces temps d'attente sont comptabilisés à l'intérieur de cette limite.

Il semble que ce paramètre soit celui qui permette d'atteindre la plus basse proportion de marché mal servi ; en effet, nous n'avons pu faire mieux que 4.9% en faisant varier les autres paramètres d'exécution mais nous pouvons obtenir 3.2% dans le cas où l'attente maximale permise est de 240 minutes, soit 4 heures.

Mais il reste des paires d'origine et destination particulières qui ne sont toujours pas desservies comme le suggère le tableau 5.13. À ce titre, prenons le cas de Toronto (YYZ) et Kelowna (YLW) qui se trouvent dans le nord de la Colombie-Britannique. Dans les données de Netline, on peut voir que certains itinéraires font escale à Calgary (YYC) où le passager y passe la nuit ; il y atterrit à 00h20 pour y décoller plus tard à

08h25, ce qui représente un temps d'attente de 8h05, soit 485 minutes. Et c'est bien au delà des valeurs que peut prendre le temps d'attente maximum dans nos simulations.

Ici, nous pouvons identifier les paires d'aéroports (YBC, YUL) et (YYZ, BGI) et (HNL, YVR) comme de bonnes candidates pour former chacune un séparateur du graphe.

Les besoins en matière d'itinéraires semblent étroitement associés aux paires d'origine et destination. Dans les données de Netline, nous n'avons pas accès à la classe des passagers (*fare class*) ou encore au type de chacune des clientèles dont il s'agit, comme gens d'affaires ou voyageurs par exemple. Il pourrait sembler une bonne idée de faire générer des itinéraires pour chacun des clientèles visées par un transporteur afin de pouvoir représenter chaque groupe de consommateurs adéquatement lors du processus d'énumération des itinéraires. Ce type de fonctionnement n'est pas implémenté dans la version des algorithmes qui ont été développés pour les présents travaux de recherche.

5.4.3 Ré-optimisations locales

Cette section présente les résultats obtenus pour la méthode de re-calculs locaux décrite dans l'algorithme 4.4.1. Nous considérons ici les résultats obtenus avec la méthode combinatoire dynamique.

À chaque itération, des segments de vols ont été retirés de l'horaire des vols et d'autres ont été ajoutés. Les modifications sont décrites en détail dans l'annexe C. Il y a trois itérations au total.

Méthode combinatoire dynamique

Les simulations numériques avec l'algorithme combinatoire avec calcul dynamique des bornes inférieures de temps de voyage entre les aéroports ont permis de recueillir les données présentées dans le tableau 5.14.

Itération	temps	itinéraires		
	(secondes)	enlevés	ajoutés	total
0	17.55	0	0	498 110
1	15.32	296	450	498 264
2	14.96	442	1 216	499 038
3	15.36	2 038	3 434	500 434

TABLEAU 5.14 Re-calculs avec la méthode combinatoire dynamique

Tel qu'attendu, l'utilisation de bornes inférieures de temps qui soient approximatives amène l'algorithme à considérer quelques itinéraires additionnels à cause d'une valeur de borne de temps inférieure qui soit moins juste qu'avec un pré-traitement qui calcule les plus courts chemins dans le réseau de transport.

Dans le contexte d'élaboration d'un horaire de vols, on pourra être appelé à effectuer plusieurs modifications mineures à un horaire initial. On peut penser alors qu'un gain en temps soit appréciable, même s'il est petit à chaque itération, car ces économies de temps s'additionnent successivement.

En observant les temps de calculs présentés dans le tableau 5.14, on est à même de penser qu'un temps initial de mise en place des structures de données de l'algorithme est important. En effet, le temps de calculs à l'itération 1 est pratiquement le même qu'à l'itération 3, mais on a pourtant retiré et ajouté beaucoup plus d'itinéraires dans le dernier cas que dans le premier. Une hypothèse réside dans le fait que l'allocation initiale de mémoire pour des structures clés, comme des tables de hachage par exemple, pourrait être la cause d'un temps constant et peu variable d'une itération à l'autre.

Par expérience, l'optimisation de code informatique dans le but de réduire les temps de calculs d'un programme représente un effort assez peu intuitif. Par exemple, on doit pouvoir mesurer le nombre d'appels à chaque fonction et le temps passé à exécuter chacune de ces fonctions. Ainsi, on peut déterminer les portions du code informatique qui représentent la plus forte proportion de temps de calculs. Mais assez souvent, la consommation de temps se trouve dans des fonctions assez banales et peu suspectées *a priori*. Une optimisation de code effectuée avec de bonnes métriques de consommation de temps pourrait contribuer à réduire les temps de calculs, que ce soit dans l'énumération d'itinéraires que dans d'autres algorithmes de résolution de problèmes.

Chapitre 6

CONCLUSION

Ce chapitre présente une conclusion des travaux de recherche présentés dans ce mémoire. Après la synthèse de la section 6.1, seront discutées certaines limitations qui pourraient être rencontrées lors d'utilisation des méthodes décrites dans les travaux dans un contexte plus industriel. Finalement, la section 6.3 propose des pistes d'améliorations futures.

6.1 Synthèse

Nous avons vu deux méthodes distinctes visant à énumérer des itinéraires de passagers dans un réseau de transport aérien. Dans chaque cas, les critères de sélection d'un itinéraire de passager ont été choisis arbitrairement mais dans l'optique de formuler des préférences d'usager d'un réseau de transport aérien. Mais ces préférences sont propres à chaque culture, chaque besoin et donc chaque clientèle.

La première méthode est élaborée à partir d'un modèle mathématique, un réseau, vastement utilisé lors de résolution de problèmes complexes. Un tel réseau permet donc d'utiliser des algorithmes connus tels que la recherche en profondeur. Il est aussi possible d'appliquer des astuces d'accélération de recherche par une heuristique comme c'est le cas dans l'algorithme A*.

La deuxième méthode a plutôt fait appel à une approche combinatoire qui considère les combinaisons d'itinéraires existants. De manière intrinsèque, cette méthode évite d'énumérer un itinéraire à plusieurs reprises. On a pu réduire les comparaisons de combinaisons en maintenant des structures de données intermédiaires qui permettent d'accéder uniquement à des sous-ensembles d'itinéraires selon certains critères pertinents au moment de produire une combinaison. Les itinéraires trouvés forment deux partitionnements principaux : un partitionnement selon l'aéroport d'origine ou un partitionnement par rapport à l'aéroport de destination. Pour chacun de ces sous-ensembles d'un partitionnement, on peut le partitionner à nouveau selon le nombre

de vols des itinéraires. On obtient donc un accès rapide aux sous-ensembles pertinents et on évite des comparaisons et calculs inutiles.

Finalement, dans un contexte d'optimisation d'un réseau de transport aérien dans son ensemble, des modifications successives à l'horaire des vols en cours de confection sont à prévoir. C'est dans cette optique que des ré-optimisations, ou re-calculs, doivent pouvoir être faits dans le voisinage des modifications à l'horaire des vols. La méthode de partitionnement permet d'isoler rapidement les répercussions d'une modification à un horaire de vols en ajustant les partitionnements. De plus, on peut générer uniquement et sans répétition les nouveaux itinéraires simplement en s'assurant de ne procéder qu'à des combinaisons d'itinéraires avec des vols qui, dans un cas ou dans l'autre, sont considérés comme nouveaux à une itération donnée.

6.2 Limitations

La méthode de recherche en profondeur a montré des limitations certaines car les algorithmes connus en recherche de chemins dans un graphe, orienté ou non, ne s'appliquent que partiellement ou mal au problème d'énumération des chemins. De plus, il est possible de penser que les algorithmes de recherche comme A* soient propices à une forte consommation de mémoire sur un système informatique. Il a été possible de réduire cette consommation en optant plutôt pour une méthode de recherche récursive ; ainsi, chaque sommet ne comporte qu'un seul jeu d'étiquettes car les étiquettes ne représentent à tout moment qu'un seul chemin en cours de confection.

L'utilisation du modèle mathématique amène à une représentation du problème qui soit arbitrairement gonflée par rapport au problème initial : pensons aux arcs d'attente qui sont ajoutées entre chaque paire de sommets qui représentent soit un décollage, soit un atterrissage d'avion à un aéroport et à un moment de la semaine. De plus, au moment de parcourir le réseau, l'algorithme est forcé de considérer les arcs d'attente, dans le cas des itinéraires, qui ne sont que de peu d'importance. En effet, on peut déduire l'information sur un temps entre un départ et une arrivée en calculant la différence entre deux temps. Lorsque plusieurs arcs d'attente successifs sont rencontrés, le traitement informatique augmente mais la valeur de l'information demeure la même.

De manière générale, le fait d'utiliser une grande quantité de mémoire amène une réduction des performances de l'algorithme. Les allocations de blocs en mémoire

par le système d'exploitation représentent un coût qui devient vite important. Des astuces de bassin de mémoire, ou *memory pool*, ont été développées dans la littérature pour réduire ce genre de dépenses de ressources. Notons que cette astuce n'a été appliquée ici que dans l'allocation d'étiquettes rattachées aux sommets du réseau dans la méthode de recherche en profondeur. De plus, une forte utilisation de la mémoire amène à des erreurs de pagination de la mémoire vive, ou *page faults*, qui affectent les temps de calcul ; si une information n'est pas présente en mémoire au moment où on souhaite y accéder, elle doit y être chargée à nouveau afin d'être accessible pour des traitements informatiques.

La méthode de recherche en profondeur a aussi le défaut de produire un même itinéraire à plusieurs reprises. Cela ne se présente tout simplement pas dans la méthode combinatoire car on procède à des combinaisons sans répétitions. La méthode combinatoire fait appel à des structures de données intermédiaires pour y récupérer des sous-ensembles d'itinéraires rapidement. Or, il faut pouvoir être en mesure de déterminer une fonction de hachage qui soit efficace pour le type de données qu'on souhaite y stocker. Dans notre cas, nous avons fait appel à une fonction de hachage sur des chaînes de caractères et il est possible d'obtenir de bons rendements dans ces cas. Mais il faut construire une chaîne de caractères pour chacun des itinéraires, ce qui représente un coût. Ce coût a été réduit en conservant localement à chaque itinéraire une telle chaîne après confection de celle-ci mais la conversion d'informations sous forme de caractères et la manipulation de chaînes de caractères représentent en soi plusieurs manipulations de mémoire qu'on néglige parfois car l'utilisateur, et voire même le programmeur, utilise souvent des objets de plus haut niveau qui cachent ces opérations multiples.

Nous avons aussi fait des pré-traitements afin d'obtenir une borne inférieure sur les temps de voyage entre chaque paire d'aéroports. Ces informations ont dû être calculées a priori mais auraient pu être disponibles, que ce soit sous forme de contraintes de marché ou de données historiques. La réelle efficacité d'une méthode de relaxation des bornes inférieures n'a pu être évaluée correctement, faute de pouvoir déterminer des métriques réelles. En fait, la qualité d'un ensemble d'itinéraires et celle d'un horaire de vols, ne sont déterminées que de manière très confidentielle par les transporteurs aériens. Comme ces données sont sensibles et qu'elles apportent directement des indications sur la concurrence et les revenus, il est difficile de pouvoir avoir une idée précise des réels impacts qu'auront une telle astuce.

La méthode combinatoire présente un avantage lorsque le partitionnement permet un gain. Si on prend le partitionnement par aéroports, d'origine ou de destination, dans un marché où il y a peu de vols par aéroport, alors ce gain s'estompe. Mais il semble raisonnable de penser qu'il y ait au moins un vol par jour à chacun des aéroports et donc le partitionnement devient avantageux.

6.3 Améliorations futures

Il semble que la méthode combinatoire représente un fort avantage par rapport à la méthode de recherche en profondeur. C'est pourquoi elle semble être un bon point de départ à partir duquel travailler dans le but d'obtenir une méthode de génération d'itinéraires qui soit performante et efficace.

Entre autres, une grande partie du secret de l'ensemble des itinéraires de passagers semble résider dans les critères de sélection des clients. Il est facile de concevoir que les besoins diffèrent d'une clientèle à une autre ; pensons aux vacanciers et aux voyageurs d'affaire par exemple. On pourrait alors penser à une catégorisation des critères d'acceptabilité selon la clientèle concernée. Cette catégorisation peut aussi être fonction des paires d'origine et destination ; les destinations vers les stations balnéaires sont moins susceptibles de répondre aux besoins d'une clientèle d'affaires. Ainsi, il pourrait être souhaitable de sélectionner une fonction d'acceptation selon la clientèle et selon la combinaison (O, D) en cours de traitement.

Aussi, on pourrait souhaiter procéder par l'utilisation d'une même fonction d'acceptabilité au moment d'exécuter une méthode de génération d'itinéraires de passagers, mais procéder à plusieurs exécutions successives en utilisant une fonction d'acceptabilité différente selon les clientèles visées par une compagnie aérienne. Par exemple, on pourrait sélectionner une fonction d'acceptabilité pour les gens d'affaires et générer les itinéraires de passagers uniquement pour les paires (O, D) de la clientèle d'affaires. Ensuite, on reprend avec une seconde fonction d'acceptation pour les vacanciers et traiter uniquement les paires (O, D) pertinentes pour cette clientèle. Et ainsi de suite.

Enfin, un profilage de l'implémentation pourrait apporter des améliorations certaines en réduisant les temps de calcul de certaines portions du code qui sont sollicitées fréquemment et qui représentent la plus grande charge de travail à effectuer par l'ordinateur. Mais ce travail relève plutôt du génie logiciel.

Références

ADELSON-VELSKII, G. et LANDIS, E. (1962). An algorithm for the organization of information. *Soviet Mathematics Doklady*, 1259–1263.

AHO, A. V., HOPCROFT, J. E. et JEFFREY D. ULLMAN, J. D. (1974). *The Design and Analysis of Computer Algorithms*. Addison Wesley Longman Publishing Co., Inc., Boston, MA, USA.

BAAJ, H. et MAHMASSANI, H. S. (1995). Hybrid route generation heuristic algorithm for the design of transit networks. *Transportation Research, Part C*, 3, 31–50.

COLDREN, G. M., KOPPELMAN, F. S., KASTURIRANGAN, K. et MUKHERJEE, A. (2003). Modeling aggregate air travel itinerary shares : Logit model development at a major u.s. airline. *Journal of Air Transport Management*, 9, 361–369.

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L. et STEIN, C. (2001). *Introduction to Algorithms*. The MIT Press, Cambridge, MA, USA, seconde édition.

DECHTER, R. et PEARL, J. (1985). Generalized best-first search strategies and the optimality of a*. *J. ACM*, 32, 505–536.

DUMAS, J. et SOUMIS, F. (2008). Passenger flow model in airline networks. *Transportation Science*, 42, 197–207. DOI : 10.1287/trsc.1070.0206.

EPPSTEIN, D. (1998). Finding the k shortest paths. *SIAM J. Computing*, 28, 652–673.

GOTTHILF, Z. et LEWENSTEIN, M. (2009). Improved algorithms for the k simple shortest paths and the replacement paths problems. *Information Processing Letters*, 109, 352–355.

GROSCHE, T. et ROTHLAUF, F. (2007). Air travel itinerary market share estimation. Working papers in information systems, Department of Information Systems, University of Mannheim.

HADJICONSTANTINOU, E. et CHRISTOFIDES, N. (1999). An efficient implementation of an algorithm for finding k shortest simple paths. *Networks*, 34, 88–101. `http://dx.doi.org/10.1002/(SICI)1097-0037(199909)34:2<88::AID-NET2>3.0.CO;2-1`.

KAINDL, H. et KAINZ, G. (1997). Bidirectional heuristic search reconsidered. *Journal of Artificial Intelligence Research*, 7, 283–317.

KLABJAN, D. (2005). *Column Generation*, Springer, New York, NY, USA, chapitre Large-scale models in the airline industry. 163–195.

KOENIG, S., LIKHACHEV, M., LIU, Y. et FURCY, D. (2004). Incremental heuristic search in ai. *AI Magazine*, 25, 99–112.

KOPPELMAN, F. S., COLDREN, G. M. et PARKER, R. A. (2008). Schedule delay impacts on air-travel itinerary demand. *Transportation Research Part B : Methodological*, 42, 263–273.

LABELLE, J. (1981). *Théorie des graphes*. Modulo, Mont-Royal, Québec, Canada.

LAWLER, E. L. (1976). *Combinatorial Optimization : Networks and Matroids*. Holt, Rinehart and Winston.

LEBEAU, L. (1984). *Génération de correspondances dans un réseau de transport aérien*. Mémoire de maîtrise, Université de Montréal.

LUBY, M. et RAGDE, P. (1989). A bidirectional shortest-path algorithm with good average-case behavior. *Algorithmica*, 4, 551–567.

LUGER, G. F. et STUBBLEFIELD, W. A. (1998). *Artificial Intelligence : Structures and Strategies for Complex Problem Solving ; Third Edition*. Addison Wesley Longman Publishing Co., Inc., Harlow, UK, troisième édition.

PEARL, J. et KORF, R. E. (1987). Search techniques. *Annual Review of Computer Science*, 2, 451–467.

PETTIE, S. (2004). A new approach to all-pairs shortest paths on real-weighted graphs. *Theoretical Computer Science*, 312, 47–74. DOI = 10.1016/S0304-3975(03)00402-X `http://doi.acm.org/10.1016/S0304-3975(03)00402-X`.

PROUSSALOGLOUA, K. et KOPPELMAN, F. S. (1999). The choice of air carrier, flight, and fare class. *Journal of Air Transport Management*, 5, 193–201.

RUSSELL, S. et NORVIG, P. (1995). *Artificial Intelligence : A Modern Approach*. Series in Artificial Intelligence. Prentice Hall, Englewood Cliffs, NJ.

VAN DER ZIJPP, N. J. et CATALANO, F. (2005). Path enumeration by finding the constrainted k-shortest paths. *Transportation Research Part B*, 39, 545–563. `doi:10.1016/j.trb.2004.07.004`.

WEN, C.-H. et LAI, S.-C. (2009). Latent class models of international air carrier choice. *Transportation Research, Part E*, 46, 211–221. DOI = 10.1016/j.rte.2009.08.004 http ://doi.acm.org/10.1016/j.rte.2009.08.004.

WU, Q. et HARTLEY, J. (2004). Accomodating user preferences in the optimization of public transport travel. *International Journal of Simulation*, 5, 12–25.

Annexe A

Descriptifs des aéroports

Voici quelques codes AITA qui représentent des aéroports, notemment ceux faisant partie de l'échantillon témoin.

Identifiant	**Aéroport**
ATL	Hartsfield-jackson Atlanta International, Atlanta, GA, États-Unis
BGI	Grantley Adams Int. (Bridgetown), Barbades
BOS	Logan Int. (Boston), Massachusetts, États-Unis
BWI	Baltimore/Washington Int., Maryland, États-Unis
CDG	Charles De Gaulle (Paris), France
HNL	Honolulu Int., Hawaï, États-Unis
IAD	Washington Dulles International, Washington, DC, États-Unis
JFK	John F Kennedy International, New York, États-Unis
LHR	London Heathrow, Angleterre
MIA	Miami International Airport, Miami, États-Unis
PDX	Portland Int., Oregon, États-Unis
PEK	Beiking Capital Int., Chine
YBC	Baie Comeau, Québec, Canada
YEG	Edmonton Int., Alberta, Canada
YQB	Québec, Québec, Canada
YHZ	Halifax Int., Nouvelle-Écosse, Canada
YLW	Kelowna Airport, Colombie-Britannique, Canada
YOW	Ottawa Macdonald-Cartier Int., Ontario, Canada
YQG	Windsor, Ontario, Canada
YQM	Greater Moncton Int., Nouveau-Brunswick, Canada
YUL	Montréal-Pierre Elliott Trudeau Int., Québec, Canada
YVR	Vancouver international, Colombie-Britannique, Canada
YWG	Winnipeg, Manitoba, Canada
YWK	Wabush, Québec, Canada
YYC	Calgary Int. Airport, Alberta, Canada
YYZ	Lester B. Pearson Int. (Toronto), Ontario, Canada
YZV	Sept-Îles, Québec, Canada

Tableau A.1 Correspondances des codes AITA de l'échantillon

Annexe B

Analyses de sensibilités, autres méthodes

B.1 Recherche en profondeur

Cette section présente les résultats obtenus pour la méthode de recherche en profondeur décrite dans l'algorithme 4.2.1 avec pré-traitement pour obtenir les valeurs de bornes inférieures de temps de voyage entre chaque paire d'aéroports.

Sur le banc de test utilisé et avec le jeu de données disponibles, notons que le pré-traitement visant à déterminer les bornes inférieures de voyage entre chaque paire d'aéroports s'est effectué en environ 35 secondes. Les temps qui figurent dans les résultats sont exprimés en secondes et représentent uniquement les temps requis par l'exécution de l'algorithme de recherche en profondeur avec utilisation de l'heuristique.

B.1.1 Facteur multiplicatif de la borne inférieure

Les variations du paramètre λ, qui est un facteur multiplicatif de la borne inférieure, donnent des résultats résumés dans les tableaux B.1 et B.2.

Critères de performance	λ				
	1.5	2.0	2.5	3.0	3.5
Temps (sec)	1405.5	2910.4	4279.1	5218.9	5879.7
Nombre d'itinéraires	202155	399618	509894	571437	606076
Couverture échantillon (%)	81.4	87.8	87.8	87.8	87.8
Couverture globale (%)	90.2	94.8	95.7	96.0	96.0
Couverture $<$ 85% (%)	60.1	61.9	57.2	53.6	52.8
Proportion de marché «mal servi» $<$ 85% (%)	18.6	8.5	6.0	5.1	5.0

TABLEAU B.1 Performances pour différentes valeurs de λ, méthode Recherche en profondeur

Paire (O, D)	λ					
	1.5	2.0	2.5	3.0	3.5	Netline
(YBC, YUL)	7	7	7	7	7	7
(YYZ, YLW)	43	347	738	844	844	142
(YWK, YOW)	6	6	6	6	6	6
(YYC, YHZ)	87	496	1240	1379	1379	172
(YYZ, BGI)	7	7	7	7	7	7
(PDX, YHZ)	103	131	131	131	131	7
(YQM, BWI)	38	38	38	38	38	21
(BOS, YUL)	39	39	39	62	112	51
(YUL, CDG)	25	174	290	340	382	46
(YOW, LHR)	67	450	678	1087	1171	83
(YEG, PEK)	72	101	101	101	101	7
(HNL, YVR)	11	11	11	11	11	11

TABLEAU B.2 Nombre d'itinéraires générés, variations du paramètre λ, méthode Recherche en profondeur

B.1.2 Nombre de segments de vol

Le paramètre indiquant le nombre maximum de segments de vol qui peuvent constituer un itinéraire de passager est k_{max}.

Nous avons condensé les critères de performance dans deux tableaux, à savoir B.3 et B.4.

Critères de performance	k_{max}			
	1	2	3	4
Temps (sec)	784.2	1157.0	2910.4	8040.5
Nombre d'itinéraires	10490	114125	399618	906542
Couverture échantillon (%)	49.6	73.4	87.8	87.8
Couverture globale (%)	69.5	94.0	94.8	94.8
Couverture $<$ 85% (%)	63.2	64.2	61.9	61.4
Proportion de marché «mal servi» $<$ 85% (%)	15.6	9.1	8.5	8.6

TABLEAU B.3 Performances pour différentes valeurs de k_{max}, méthode Recherche en profondeur

Paire (O, D)	k_{max}				
	1	2	3	4	Netline
(YBC, YUL)	7	7	7	7	7
(YYZ, YLW)	1	192	347	347	142
(YWK, YOW)	0	0	6	105	6
(YYC, YHZ)	7	190	496	507	172
(YYZ, BGI)	7	7	7	7	7
(PDX, YHZ)	0	0	131	575	7
(YQM, BWI)	0	7	38	207	21
(BOS, YUL)	39	39	39	39	51
(YUL, CDG)	14	57	174	179	46
(YOW, LHR)	7	102	450	563	83
(YEG, PEK)	0	14	101	195	7
(HNL, YVR)	11	11	11	11	11

TABLEAU B.4 Nombre d'itinéraires générés, variations du paramètre k_{max}, méthode Recherche en profondeur

B.1.3 Temps minimum de connexion

Les deux tableaux B.5 et B.6 montrent les performances obtenues en variant le paramètre t_{min} parmi les valeurs $\{30, 60, 90\}$.

Critères de performance	t_{min}		
	30	60	90
Temps (sec)	2910.4	1738.9	1263.7
Nombre d'itinéraires	399618	244177	136275
Couverture échantillon (%)	87.8	70.3	65.9
Couverture globale (%)	94.8	90.1	82.8
Couverture $<$ 85% (%)	61.9	59.8	50.6
Proportion de marché «mal servi» $<$ 85% (%)	8.5	18.8	29.4

TABLEAU B.5 Performances pour différentes valeurs de t_{min}, méthode Recherche en profondeur

Paire (O, D)	t_{min}			
	30	60	90	Netline
(YBC, YUL)	7	7	7	7
(YYZ, YLW)	347	170	110	142
(YWK, YOW)	6	0	0	6
(YYC, YHZ)	496	236	110	172
(YYZ, BGI)	7	7	7	7
(PDX, YHZ)	131	68	42	7
(YQM, BWI)	38	34	13	21
(BOS, YUL)	39	39	39	51
(YUL, CDG)	174	114	66	46
(YOW, LHR)	450	285	111	83
(YEG, PEK)	101	73	33	7
(HNL, YVR)	11	11	11	11

TABLEAU B.6 Nombre d'itinéraires générés, variations du paramètre t_{min}, méthode Recherche en profondeur

B.1.4 Attente maximale pour connexion

Les tableaux B.7 et B.8 comportent les données pour les calculs avec différentes valeurs du paramètre t_{max}.

Critères de performance	t_{max}				
	120	150	180	210	240
Temps (sec)	1703.7	2225.0	2910.4	3494.7	4203.2
Nombre d'itinéraires	228672	313205	399618	470842	536862
Couverture échantillon (%)	78.4	87.5	87.8	90.2	92.4
Couverture globale (%)	89.0	92.8	94.8	95.4	95.7
Couverture $<$ 85% (%)	60.0	60.8	61.9	62.7	63.3
Proportion de marché «mal servi» $<$ 85% (%)	20.1	12.0	8.5	7.5	7.0

TABLEAU B.7 Performances pour différentes valeurs de t_{max}, méthode Recherche en profondeur

Paire (O, D)	t_{max}					
	120	150	180	210	240	Netline
(YBC, YUL)	7	7	7	7	7	7
(YYZ, YLW)	266	313	347	360	360	142
(YWK, YOW)	6	6	6	6	6	6
(YYC, YHZ)	380	463	496	531	531	172
(YYZ, BGI)	7	7	7	7	7	7
(PDX, YHZ)	49	89	131	179	221	7
(YQM, BWI)	25	31	38	57	73	21
(BOS, YUL)	39	39	39	39	39	51
(YUL, CDG)	117	140	174	188	202	46
(YOW, LHR)	272	374	450	518	559	83
(YEG, PEK)	34	68	101	131	158	7
(HNL, YVR)	11	11	11	11	11	11

TABLEAU B.8 Nombre d'itinéraires générés, variations du paramètre t_{max}, méthode Recherche en profondeur

B.2 Recherche en profondeur dynamique

Contrairement à la section B.1, le pré-traitement pour obtenir des bornes inférieures de temps de voyage n'a pas été requis. Les temps montrés ici, et exprimés en secondes, sont ceux obtenus par exécution de la recherche en profondeur seule.

B.2.1 Facteur multiplicatif de la borne inférieure

Les variations du paramètre de facteur multiplicatif de la borne inférieure donnent des résultats résumés dans les tableaux B.9 et B.10.

Critères de performance	λ				
	1.5	2.0	2.5	3.0	3.5
Temps (sec)	669.3	1400.6	1847.8	2174.6	2392.9
Nombre d'itinéraires	89989	141884	167484	182480	191852
Couverture échantillon (%)	59.3	59.8	59.8	59.8	59.8
Couverture globale (%)	78.7	79.9	80.3	80.4	80.4
Couverture $<$ 85% (%)	60.7	60.7	59.8	59.5	59.5
Proportion de marché «mal servi» $<$ 85% (%)	16.1	13.1	12.1	11.7	11.7

TABLEAU B.9 Performances pour différentes valeurs de λ, méthode Recherche en profondeur dynamique

Paire (O, D)	λ					
	1.5	2.0	2.5	3.0	3.5	Netline
(YBC, YUL)	7	7	7	7	7	7
(YYZ, YLW)	1	1	1	1	1	142
(YWK, YOW)	0	0	0	0	0	6
(YYC, YHZ)	87	489	1227	1359	1359	172
(YYZ, BGI)	7	7	7	7	7	7
(PDX, YHZ)	0	0	0	0	0	7
(YQM, BWI)	1	1	1	1	1	21
(BOS, YUL)	39	39	39	39	39	51
(YUL, CDG)	14	14	14	14	14	46
(YOW, LHR)	32	64	72	72	72	83
(YEG, PEK)	72	101	101	101	101	7
(HNL, YVR)	11	11	11	11	11	11

TABLEAU B.10 Nombre d'itinéraires générés, variations du paramètre λ, méthode Recherche en profondeur dynamique

B.2.2 Nombre de segments de vol

Selon le nombre maximum de segments de vol qu'on admettra dans un itinéraire, on aura les performances indiquées dans les tableaux B.11 et B.12.

Critères de performance	k_{max}			
	1	2	3	4
Temps (sec)	101.7	307.7	1400.6	4706.3
Nombre d'itinéraires	10490	48408	141884	292093
Couverture échantillon (%)	49.6	60.2	59.8	59.8
Couverture globale (%)	69.5	79.6	79.9	79.9
Couverture $<$ 85% (%)	63.2	61.4	60.7	61.3
Proportion de marché «mal servi» $<$ 85% (%)	15.6	13.1	13.1	13.0

TABLEAU B.11 Performances pour différentes valeurs de k_{max}, méthode Recherche en profondeur dynamique

Paire (O, D)	k_{max}				
	1	2	3	4	Netline
(YBC, YUL)	7	7	7	7	7
(YYZ, YLW)	1	1	1	1	142
(YWK, YOW)	0	0	0	0	6
(YYC, YHZ)	7	190	489	500	172
(YYZ, BGI)	7	7	7	7	7
(PDX, YHZ)	0	0	0	0	7
(YQM, BWI)	0	1	1	0	21
(BOS, YUL)	39	39	39	39	51
(YUL, CDG)	14	14	14	14	46
(YOW, LHR)	7	32	64	64	83
(YEG, PEK)	0	14	101	195	7
(HNL, YVR)	11	11	11	11	11

TABLEAU B.12 Nombre d'itinéraires générés, variations du paramètre k_{max}, méthode Recherche en profondeur dynamique

B.2.3 Temps minimum de connexion

Pour les différentes valeurs de temps minimum requis pour compléter une connexion dans un itinéraire, nous avons les performances telles que montrées dans les tableaux B.13 et B.14.

Critères de performance	t_{min}		
	30	60	90
Temps (sec)	1400.6	744.2	490.9
Nombre d'itinéraires	141884	98225	62695
Couverture échantillon (%)	59.8	59.3	58.6
Couverture globale (%)	79.9	77.7	74.8
Couverture $<$ 85% (%)	60.7	59.5	54.2
Proportion de marché «mal servi» $<$ 85% (%)	13.1	17.8	21.9

TABLEAU B.13 Performances pour différentes valeurs de t_{min}, méthode Recherche en profondeur dynamique

Paire (O, D)	t_{min}			
	30	60	90	Netline
(YBC, YUL)	7	7	7	7
(YYZ, YLW)	1	1	1	142
(YWK, YOW)	0	0	0	6
(YYC, YHZ)	489	229	110	172
(YYZ, BGI)	7	7	7	7
(PDX, YHZ)	0	0	0	7
(YQM, BWI)	1	1	0	21
(BOS, YUL)	39	39	39	51
(YUL, CDG)	14	14	14	46
(YOW, LHR)	64	37	19	83
(YEG, PEK)	101	73	33	7
(HNL, YVR)	11	11	11	11

TABLEAU B.14 Nombre d'itinéraires générés, variations du paramètre t_{min}, méthode Recherche en profondeur dynamique

B.2.4 Attente maximale pour connexion

En faisant varier le paramètre de temps d'attente maximal entre deux connexions d'un itinéraire, on obtient les tableaux B.15 et B.16.

Critères de performance	t_{max}				
	120	150	180	210	240
Temps (sec)	621.9	931.7	1400.6	1665.9	2062.8
Nombre d'itinéraires	81776	111906	141884	168268	191307
Couverture échantillon (%)	51.2	59.7	59.8	59.8	59.8
Couverture globale (%)	77.8	79.2	79.9	80.1	80.2
Couverture $<$ 85% (%)	61.5	60.7	60.7	60.7	60.8
Proportion de marché «mal servi» $<$ 85% (%)	17.9	14.4	13.1	12.9	12.7

TABLEAU B.15 Performances pour différentes valeurs de t_{max}, méthode Recherche en profondeur dynamique

Paire (O, D)	t_{max}					
	120	150	180	210	240	Netline
(YBC, YUL)	7	7	7	7	7	7
(YYZ, YLW)	1	1	1	1	1	142
(YWK, YOW)	0	0	0	0	0	6
(YYC, YHZ)	373	456	489	524	524	172
(YYZ, BGI)	7	7	7	7	7	7
(PDX, YHZ)	0	0	0	0	0	7
(YQM, BWI)	1	1	1	1	1	21
(BOS, YUL)	39	39	39	39	39	51
(YUL, CDG)	14	14	14	14	14	46
(YOW, LHR)	58	58	64	64	78	83
(YEG, PEK)	34	68	101	131	158	7
(HNL, YVR)	11	11	11	11	11	11

TABLEAU B.16 Nombre d'itinéraires générés, variations du paramètre t_{max}, méthode Recherche en profondeur dynamique

B.3 Méthode combinatoire

Les résultats présentés dans cette section sont ceux obtenus par exécution d'une implémentation de l'algorithme 4.3.1 avec pré-traitement pour obtenir les bornes inférieures sur les temps de calculs. Pour chacun des temps mentionnés dans cette section, il faut considérer qu'un temps de 35 secondes a été requis à titre de pré-traitement afin d'obtenir des valeurs de bornes inférieures de temps pour l'heuristique.

B.3.1 Facteur multiplicatif de la borne inférieure

Les tableaux B.17 et B.18 présentent les critères de performance selon la variation du paramètre λ.

Critères de performance	λ				
	1.5	2.0	2.5	3.0	3.5
Temps (sec)	8.9	13.2	15.1	15.9	16.6
Nombre d'itinéraires	164643	344625	452555	512337	553946
Couverture échantillon (%)	81.4	87.8	87.8	87.8	87.8
Couverture globale (%)	90.1	94.7	95.7	96.0	96.0
Couverture $<$ 85% (%)	60.3	62.2	57.5	53.8	52.9
Proportion de marché «mal servi» $<$ 85% (%)	18.7	8.6	6.0	5.1	5.0

TABLEAU B.17 Performances pour différentes valeurs de λ, méthode Combinatoire

Paire (O, D)	λ					
	1.5	2.0	2.5	3.0	3.5	Netline
(YBC, YUL)	7	7	7	7	7	7
(YYZ, YLW)	43	347	738	844	844	142
(YWK, YOW)	6	6	6	6	6	6
(YYC, YHZ)	87	496	1240	1379	1379	172
(YYZ, BGI)	7	7	7	7	7	7
(PDX, YHZ)	103	131	131	131	131	7
(YQM, BWI)	11	32	38	38	38	21
(BOS, YUL)	39	39	39	62	112	51
(YUL, CDG)	25	71	84	111	168	46
(YOW, LHR)	67	137	198	527	618	83
(YEG, PEK)	14	20	39	66	72	7
(HNL, YVR)	11	11	11	11	11	11

TABLEAU B.18 Nombre d'itinéraires générés, variations du paramètre λ, méthode Combinatoire

B.3.2 Nombre de segments de vol

Les tableaux B.19 et B.20 présentent les critères de performances en fonction des valeurs que peut prendre le paramètre k_{max}.

Critères de performance	k_{max}			
	1	2	3	4
Temps (sec)	0.0	2.7	13.2	32.5
Nombre d'itinéraires	10490	114125	344625	589460
Couverture échantillon (%)	49.6	73.4	87.8	87.8
Couverture globale (%)	69.5	94.0	94.7	94.7
Couverture $<$ 85% (%)	63.2	64.2	62.2	62.0
Proportion de marché «mal servi» $<$ 85% (%)	15.6	9.1	8.6	8.6

TABLEAU B.19 Performances pour différentes valeurs de k_{max}, méthode Combinatoire

Paire (O, D)	k_{max}				
	1	2	3	4	Netline
(YBC, YUL)	7	7	7	7	7
(YYZ, YLW)	1	192	347	347	142
(YWK, YOW)	0	0	6	105	6
(YYC, YHZ)	7	190	496	507	172
(YYZ, BGI)	7	7	7	7	7
(PDX, YHZ)	0	0	131	505	7
(YQM, BWI)	0	7	32	32	21
(BOS, YUL)	39	39	39	39	51
(YUL, CDG)	14	57	71	71	46
(YOW, LHR)	7	102	137	137	83
(YEG, PEK)	0	14	20	20	7
(HNL, YVR)	11	11	11	11	11

TABLEAU B.20 Nombre d'itinéraires générés, variations du paramètre k_{max}, méthode Combinatoire

B.3.3 Temps minimum de connexion

Les différentes valeurs que peut prendre le paramètre t_{min} influent sur les critères de performance des itinéraires de passagers tels que montré dans les tableaux B.21 et B.22.

Critères de performance	t_{min}		
	30	60	90
Temps (sec)	13.2	9.9	7.1
Nombre d'itinéraires	344625	211916	120181
Couverture échantillon (%)	87.8	70.3	65.9
Couverture globale (%)	94.7	90.1	82.8
Couverture $<$ 85% (%)	62.2	59.9	50.8
Proportion de marché «mal servi» $<$ 85% (%)	8.6	18.8	29.3

TABLEAU B.21 Performances pour différentes valeurs de t_{min}, méthode Combinatoire

Paire (O, D)	t_{min}			
	30	60	90	Netline
(YBC, YUL)	7	7	7	7
(YYZ, YLW)	347	170	110	142
(YWK, YOW)	6	0	0	6
(YYC, YHZ)	496	236	110	172
(YYZ, BGI)	7	7	7	7
(PDX, YHZ)	131	68	42	7
(YQM, BWI)	32	28	7	21
(BOS, YUL)	39	39	39	51
(YUL, CDG)	71	50	46	46
(YOW, LHR)	137	88	54	83
(YEG, PEK)	20	7	7	7
(HNL, YVR)	11	11	11	11

TABLEAU B.22 Nombre d'itinéraires générés, variations du paramètre t_{min}, méthode Combinatoire

B.3.4 Attente maximale pour connexion

En faisant varier le paramètre t_{max}, on obtient des valeurs de performances telles que montrées dans les tableaux B.23 et B.24.

Critères de performance	t_{max}				
	120	150	180	210	240
Temps (sec)	9.2	11.1	13.2	14.4	15.6
Nombre d'itinéraires	203483	274665	344625	398616	446343
Couverture échantillon (%)	78.4	87.5	87.8	90.2	92.4
Couverture globale (%)	88.9	92.8	94.7	95.3	95.6
Couverture $<$ 85% (%)	60.1	61.0	62.2	63.1	63.6
Proportion de marché «mal servi» $<$ 85% (%)	20.1	12.1	8.6	7.6	7.0

TABLEAU B.23 Performances pour différentes valeurs de t_{max}, méthode Combinatoire

Paire (O, D)	t_{max}					
	120	150	180	210	240	Netline
(YBC, YUL)	7	7	7	7	7	7
(YYZ, YLW)	266	313	347	360	360	142
(YWK, YOW)	6	6	6	6	6	6
(YYC, YHZ)	380	463	496	531	531	172
(YYZ, BGI)	7	7	7	7	7	7
(PDX, YHZ)	49	89	131	179	221	7
(YQM, BWI)	25	25	32	37	49	21
(BOS, YUL)	39	39	39	39	39	51
(YUL, CDG)	46	50	71	71	85	46
(YOW, LHR)	89	124	137	172	186	83
(YEG, PEK)	13	20	20	20	25	7
(HNL, YVR)	11	11	11	11	11	11

TABLEAU B.24 Nombre d'itinéraires générés, variations du paramètre t_{max}, méthode Combinatoire

Annexe C

Itérations lors de recalculs

Voici les modifications à l'horaire de vols initialement décrit par les données de Netline qui ont été utilisées lors des essais numériques avec les méthodes de re-calcul. Il y a trois (3) itérations décrites ci-après.

1. À l'itération 0, nous utilisons l'horaire des vols tel que décrit dans les données de Netline, notre jeu de données complet ;
2. À l'itération 1, on retire les segments de vol suivants :
   - AC 382, partant de Montréal-Trudeau (YUL) à 09h55 heure locale et atterrissant à Boston (BOS) à 11h10, heure locale, du lundi au vendredi.

   et on ajoute les segments de vol suivants :
   - un vol fictif AC 9991a de Montréal-Trudeau (YUL) à 14h45 vers Boston (BOS) à 15h55 du lundi au dimanche ;
   - un vol fictif AC 9991b de Montréal-Trudeau (YUL) à 10h25 vers Boston (BOS) à 11h50
   -
3. À l'itération 2, on retire les segments de vol suivants :
   - AC 389, partant de Boston (BOS) à 19h40 et atterrissant à Montréal-Trudeau (YUL) à 20h50, du lundi au vendredi inclusivement ;
   - AC 548 partant de Vancouver (YVR) à 14h15 et atterrissant à New York (JFK) à 22h29 heure locale, du lundi au dimanche ;
   - AC 351 en partant de Washington (IAD) à 17h20 et atterrissant à Montréal-Trudeau (YUL) à 19h00, du lundi au dimanche.

   et on ajoute les segments de vol suivants :
   - un vol fictif AC 9992a de Vancouver (YVR) à 14h15 vers New York (JFK) à 22h29, du lundi au dimanche ;
   - un vol fictif AC 9992b de Boston (BOS) à 19h40 vers Montréal (YUL) à 20h50, du lundi au dimanche ;

- un vol fictif AC 9992c de Washington (IAD) à 17h20 vers Montréal-Trudeau (YUL) à 19h00, du lundi au dimanche ;

4. À l'itération 3, les vols suivants sont retirés de l'horaire des vols :
   - AC 346 de Montréal-Trudeau (YUL) à 06h15 vers Washington (IAD) à 08h02, du lundi au dimanche ;
   - AC 383 de Boston (BOS) à 11h40 vers Montréal-Trudeau (YUL) à 12h50, du lundi au vendredi ;
   - AC 548 de Vancouver (YVR) à 14h15 vers New York (JFK) à 22h29, du lundi au dimanche ;
   - AC 549 de New York (JFK) à 07h45 vers Vancouver (YVR) à 10h42, du lundi au dimanche ;
   - AC 930 de Montréal-Trudeau (YUL) à 08h30 vers Miami (MIA) à 11h59, du lundi au dimanche ;
   - AC 933 de Miami (MIA) à 12h50 vers Montréal-Trudeau (YUL) à 16h10, du lundi au dimanche ;
   - AC 1102 de Toronto (YYZ) à 10h00 vers Atlanta (ATL) à 12h15, toute la semaine sauf le samedi ;
   - AC 8643 de Atlanta (ATL) à 10h05 vers Toronto (YYZ) à 12h25, du lundi au dimanche ;

   et on ajoute les vols suivants :
   - un vol fictif AC 9993a de Montréal-Trudeau (YUL) à 08h50 vers Miami (MIA) à 12h19, du lundi au dimanche ;
   - un vol fictif AC 9993b de Montréal-Trudeau (YUL) à 07h30 vers Miami (MIA) à 10h59, du lundi au dimanche ;
   - un vol fictif AC 9993c de Miami (MIA) à 12h25 vers Montréal (YUL) à 15h45, du lundi au dimanche ;
   - un vol fictif AC 9993d de Toronto (YYZ) à 10h00 vers Atlanta (ATL) à 12h15, du lundi au dimanche ;
   - un vol fictif AC 9993e de Atlanta (ATL) à 10h05vers Toronto (YYZ) à 12h25, du lundi au dimanche ;
   - un vol fictif AC 9993f de Boston (BOS) à 11h15 vers Montréal (YUL) à 12h25, chaque jour de la semaine ;
   - un vol fictif AC 9993g de New York (JFK) à 07h05 vers Vancouver (YVR) à 10h02, chaque jour de la semaine ;

- un vol fictif AC 9993h de New York (JFK) à 08h25 vers Vancouver (YVR) à 11h22, du lundi au dimanche ;
- un vol fictif AC 9993i de Vancouver (YVR) à 14h45 vers New York (JFK) à 22h59, chaque jour de la semaine ;
- un vol fictif AC 9993j de Montréal-Trudeau (YUL) à 06h40 vers Washington (IAD) à 08h27, du lundi au dimanche.